Terrain Modelling with Triangle Based Free-Form Surfaces

Dissertation
zur
Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)
vorgelegt der
Mathematisch-naturwissenschaftlichen Fakultät
der
Universität Zürich
von

Marco Hugentobler
von Mönchaltorf

Promotionskomitee
Prof. Dr. Robert Weibel (Vorsitz)
Dr. Ross Purves
Dr. Bernhard Schneider
Dr. Marc van Kreveld

Zürich 2004

Abstract

The shape of the terrain surface is important for many GIS applications. Currently, digital terrain modelling is usually done by interpolation of the input data to a regular grid and applying an algorithm to obtain the required terrain derivatives. As several algorithms may use different implicit surfaces, this approach may introduce inconsistency. Furthermore, interpolation is performed twice and because each interpolation step introduces uncertainty, this may not be optimal. An alternative approach is to specify a continuous surface explicitly and to derive all terrain information from this surface.

Reconstruction of continuous surfaces from discrete input data is underdefined. Therefore it is important to use all available information to reduce shape uncertainty. This includes the removal of artifacts because an interpolation artifact can be seen as a surface behaviour which is, based about knowledge about terrain, implausible.

Because of their intuitive shape control and their local nature, methods from Computer Aided Geometric Design (CAGD) seem to be well suited to continuous terrain modelling. This thesis examines the suitability of triangle-based Coons patches and Clough-Tocher Bézier splines for terrain modelling. The basic idea of Coons patches is to specify a network of boundary curves and to find a surface which passes through these curves. Clough-Tocher Bézier splines, in contrast, use control points to describe the shape of the surface. Specifically, the possibilities to include linear information and the occurence and removal of artifacts with these methods are examined.

In terrain modelling it is often important that linear elements are on the surface. Examples of such elements are contours, lake boundaries or valleys. Linear elements can be considered in triangle-based surface by using constrained Delaunay triangulations. However, across some linear elements, for instance sharp ridges, lake boundaries or road edges, it is desirable that the digital surface changes its gradient aprupt. Therefore, extensions to include breaklines in otherwise continuously differentiable surfaces, are developed for Coons patches and Clough-Tocher splines. In both kinds of surfaces, different kinds of artifacts frequently occur. The usage of linear interpolated cross-derivatives, for instance, creates well visible artifacts. To soften these artifacts, a method to make the transitions between Clough-Tocher triangles smoother is introduced. Cubic surfaces may undulate in long and thin triangles. In this thesis, the Ruppert algorithm has been applied to alter the triangular mesh such that small angles in long thin triangles disappear.

The methods developed in the thesis are evaluated by means of artificial surfaces and field data. On the artificial surfaces as well as in the test area, the cubic free-form surfaces show a considerably improved prediction of elevations in comparison to linear interpolation because the cubic interpolators model the curvature of the terrain. The consideration of breaklines also improves the the predictions. The differences between the cubic interpolators are small in all experiments because of two reasons. First, although based on different mathematical formulations, Clough-Tocher Bézier splines and Coons patches don't show large differences if they have the same degree and the same cross-boundary derivative function. Second, the smoothing of the Clough-Tocher splines has an impact on the visual appearance of a surface, but the numerical effect of this extension is small in the experiments.

Zusammenfassung

Die Form der Geländeoberfläche spielt für viele GIS-Anwendungen eine wichtige Rolle. Meistens wird in der Geländemodellierung aus den Inputdaten ein regelmässiges Gitter interpoliert. Dann wird auf das Gitter ein Algorithmus angewendet, welcher die gewünschte Geländeinformation extrahiert. Da solche Algorithmen verschiedene implizite Oberflächen verwenden kann so ein Vorgehen Inkonsistenzen zur Folge haben, falls für eine räumliche Modellierung mehrere Arten von Geländeinformation verwendet werden. Ausserdem wird so zweimal interpoliert, was auch nicht wünschenswert ist, da jeder Interpolationsschritt mit Unsicherheiten behaftet ist. Ein alternatives Vorgehen ist daher, eine kontinuierliche Oberfläche explizit zu spezifizieren und alle Geländeinformation direkt daraus abzuleiten.

Die Rekonstruktion einer kontinuierlichen Oberfläche aus Punkt- und Liniendaten ist unterdefiniert. Es ist daher bei der Rekonstruktion wichtig, alle zur Verfügung stehende Information zu nutzen, damit die durch die Interpolation verursachten Unsicherheiten möglichst klein sind. Da man ein Interpolationsartefakt als ein Verhalten der Oberfläche anschauen kann, welches aufgrund des Vorwissens über Geomorphologie sehr unwahrscheinlich ist, schliesst dies das Vermeiden von Artefakten mit ein.

Methoden aus dem Computer Aided Geometric Design (CAGD) scheinen für das Spezifizieren von kontinuierlichen Oberflächen sehr geeignet zu sein, da sie eine intuitive Beeinflussung der Oberflächenform erlauben. Diese Dissertion beschäftigt sich mit der Frage, inwiefern sich kubische Coons patches und Clough-Tocher Béziersplines für die digitale Geländemodellierung eignen. Das Grundprinzip von Coons patches besteht darin, aus einem Netzwerk von Randkurven eine geeignete Oberfläche zu finden, welche diese Kurven interpoliert. Clough-Tocher Béziersplines dagegen verwenden Kontrollpunkte, welche die Oberfläche anziehen und so deren Form beeinflussen. Im speziellen wird untersucht, wie lineare Information in diesen Oberflächen berücksichtigt werden kann sowie welche Artefakte auftreten und wie sie vermieden werden können.

In der Geländemodellierung ist es oftmals wichtig, dass lineare Elemente auf der Oberfläche sind. Beispiele dafür sind Höhenlinien, Seekonturen oder Tallinien. Linien können in dreiecksbasierten Oberflächen berücksichtigt werden, indem gezwungene Delaunaytriangulationen verwendet werden. Bei einigen Linienelementen, z. B. scharfen Graten, Seeufern oder Strassenrändern, kann es aber wünschenswert sein, dass die Oberfläche die Richtung aprupt ändert. Daher werden sowohl für die Coons patches als auch für die Clough-Tocher Bézier splines Erweiterungen entwickelt, welche die Berücksichtigung von Bruchkanten in den ansonsten kontinuierlich differenzierbaren Oberflächen ermöglichen. In beiden Arten von Oberflächen gibt es eine Reihe von Artefakten, welche immer wieder auftreten. Die Verwendung von linear interpolierten Ableitungen senkrecht zu den Randkurven führt zu gut sichtbaren Artefakten. Um diese abzuschwächen wird eine Methode verwendet, um die Übergänge zwischen zwei Bézierdreiecken möglichst glatt zu machen. Kubische Oberflächen können in langen und spitzen Dreiecken ausschwingen. Daher wird der Algorithmus von Ruppert verwendet, welcher durch Einfügen zusätzlicher Punkte kleine Winkel in der Triangulation verhindert.

Die im Rahmen dieser Dissertation entwickelten Methoden werden mit Hilfe von künstlichen Oberflächen und mit Daten aus dem Feld evaluiert. Sowohl auf den künstlichen Oberflächen wie auch im Testgelände schneiden die kubischen Oberflächen verglichen mit der linearen Interpolation erheblich besser ab, weil sie auch die Krümmung des Geländes berücksichtigen. Auch die Berücksichtigung von Bruchkanten führt zu einer Verbesserung des Interpolationsergebnisses. Die Unterschiede zwischen den verschiedenen kubischen Interpolatoren sind bei allen Versuchen sehr klein. Zum einen

sind die Unterschiede zwischen Coons patches und Clough-Tocher Bézier splines sehr klein, falls beide kubisch sind und dieselben Ableitungsfunktionen senkrecht zu den Randkurven haben. Zum anderen hat die Glättung der Clough-Tocher splines wohl einen Einfluss auf die visuelle Erscheinung der Oberflächen, die numerischen Effekte sind aber in allen Experimenten klein.

Danksagung

- Prof. Dr. Robert Weibel für die Möglichkeit, eine Arbeit in seiner Abteilung zu schreiben. Trotz randvollem Terminkalender fand er immer wieder Zeit für eine Diskussion oder um die Kapitel dieser Diss durchzugehen.
- Dr. Ross Purves für die engagierte Betreuung, die vielen Ideen, welche in diese Arbeit einflossen, sowie seinen unerschütterlichen Optimismus, dass diese Arbeit zu einem guten Ende kommt.
- Dr. Bernhard Schneider, welcher es immer wieder verstand, mich mit seinem Enthusiasmus für Oberflächen- und Geländemodellierung anzustecken und mir jederzeit mit Rat und Tat zur Seite stand.
- Alistair Edwards für die Hilfe beim Vermessen des Testgeländes.
- Den Mitglieder der DTM-Gruppe, Felix Hebeler, Syed Awase Khirni, Daria Martinoni, Dani Wirz und Bisheng Yang für das gute Arbeitsklima im L94.
- Meiner Frau Nathalie und meiner Tochter Nora, welche es mit viel Geduld ertrugen, wenn ich mal wieder zusätzlich an einem Wochenende an dieser Diss arbeitete.
- Meinen Eltern sowie allen Grosseltern von Nora. Oftmals schauten sie zu meiner Tochter, wenn ich mal wieder an einem Abend oder an einem Wochenende an die Uni ging.
- Dem Schweizerischen Nationalfonds, welcher diese Dissertation als Teil des Projektes 'Management of Metainformation and Uncertainties in Digital Terrain Modelling' (Projekt nr. 2000-059578.99) finanzierte.
- Der GIS-Fachstelle des Kantons Zug, welches das digitale Terrainmodell des Kantons Zug sowie Fixpunktkoordinaten zur Verfügung stellte.
- Dem Bundesamt fuer Landestopographie für die Erlaubnis, Höhendaten aus dem Zuestollgebiet manuell herauszuschreiben und für den Demodatensatz Albis.

Contents

Chapter 1

Introduction

1.1 Digital modelling of continuous terrain surfaces

1.1.1 Digital terrain modelling

The terrain surface is important for a large number of applications in science and engineering. To mention only a few, terrain is crucial in hydrological modelling, placement of anntenas, production of orthophotos, ice sheet modelling and forest fire prediction (Weibel and Heller, 1990).

Digital modelling of the terrain surface allows for the calculation of many derived products besides elevation. Elevation, slope, aspect, curvature, the visible area from a point and fill or cut volumes are only a few examples of the large number of derivatives which can be generated from a digital terrain surface. Table 1.1 provides a list of derivatives useful in hydrological modelling.

Several data sources are popular for digital terrain models:

- contours digitised from topographic maps are a relatively cheap data source. Most national mapping agencies sell digitised contour data and/ or an interpolated grid model derived from it;
- analytical photogrammetry is carried out interactively and allows the measurement of selected points and linear features from a pair of aerial images. In wooded areas, photogrammetry can not be applied due to obstructed visibility of the terrain surface. Normally, in such areas, another data source is used (e. g. digitised contours).
- digital photogrammetry provides automated selection of data points which are usually arranged in a regular grid
- laser scanning is a relatively new technique which allows for the collection of very dense terrain datasets. In contrast to photogrammetry, laser scanning (or LIDAR) can also be used in wooded areas. Nevertheless, the distinction between surface and terrain is often difficult (Pfeifer et al., 2001).
- SAR interferometry is another source of terrain data which allows for the creation of regular grids. A famous example covering the whole world is the dataset obtained from the Shuttle Radar Topography Mission Pac (2000). As raw SAR digital elevation models (DEMs) have occasional large errors and random elevation errors, careful filtering and interpolation of such data is required (Hutchinson and Gallant, 2000).

Table 1.1: Primary topographic attributes in hydrological modelling [Beven and Moore 1991].

Attribute	**Definition**
altitude	elevation
upslope height	mean height of upslope area
aspect	slope azimuth
slope	gradient
upslope slope	mean slope of upslope area
dispersal slope	mean slope of dispersal area
catchment slope	average slope over the catchment
upslope area	catchment area above a short length of contour
dispersal area	area downslope from a short length of contour
catchment area	area draining to catchment outlet
specific catchment area	upslope area per unit width of contour
flow path length	maximum distance of water flow to a point in the catchment
upslope length	mean length of flow paths to a point in the catchment
dispersal length	distance from a point in the catchment to the outlet
catchment length	distance from highest point to outlet
profile curvature	slope profile curvature
plan curvature	contour curvature

Terrain modelling is usually carried out the following way: First, the continuous terrain surface is discretised using one of the measurement techniques above. The terrain data is then interpolated to a regular grid. To derive topographic information from this set of discrete data points, standard GIS software provides a large number of algorithms. As derivatives are not defined for discrete data points, however, these algorithms usually specify continuous surfaces implicitly (Schneider, 2001b; Wood, 1998). An example for this is the calculation of curvature in an elevation grid by the method of Zevenbergen and Thorne (1987). This algorithm implicitly specifies a local quadratic surface, centred at the point for which curvature has to be calculated. In some cases, there is no specification of a continuous surface at all, e. g. when using the single flow direction method (O'Callaghan and Mark, 1984; Gallant and Wilson, 2000) to derive flowlines in a regular grid.

1.1.2 Shortcomings of existing approaches

The specification of surfaces by the algorithms to derive terrain information causes various problems in terrain modelling.

If different types of terrain derivatives are used within one modelling task, the derivatives may be based on different surfaces and thus introduce inconsistencies (Martinoni, 2002). If, for instance, slope and curvature are used for a spatial model and their values are calculated using the ARC/INFO functions SLOPE and CURVATURE, the two values are based on different implicit surfaces. While the calculation of slope is based on finite differences, curvature is calculated using piecewise quadratic surfaces (ESRI, 2002; Horn, 1981; Zevenbergen and Thorne, 1987). Another example for such an inconsistency is the usage of flow lines, calculated using the D8-Algorithm (O'Callaghan and Mark, 1984), and aspect, calculated with finite differences (ESRI, 2002). In this case, the inconsistency is obvious, because the directions of the flow lines and the aspect values at the data points do not necessarily match.

It is hard for a user of terrain information to figure out the shape of implicitly specified surfaces. Explicitly specifying surfaces in contrast forces a user to think in terms of surfaces and not in terms of discrete points.

The algorithms which implicitly specify surfaces are often simple and do not use the available information about the shape of the terrain well. Figure 1.1 shows an example of a surface consisting of local biquadratic surfaces. Because this approach does not consider the relations between adjacent cells, the surface has a lot of gaps.

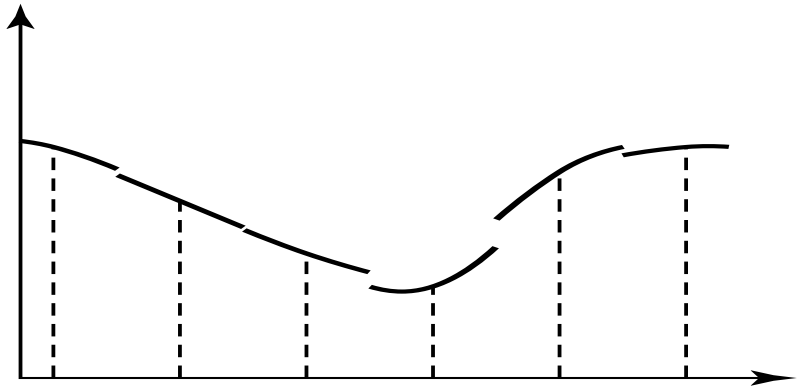

Figure 1.1: Surface consisting of local biquadratic surfaces.

1.1.3 Proposed approach

This thesis builds on the hypothesis that the problems mentioned above can be avoided by modelling terrain explicitely as a mathematical function. Because all terrain information derived would be based on this function, inconsistency would not be a problem anymore. As caves and overhangs are relatively rare in nature, the restriction of a function to have only one value for a location is considered to be appropriate for most applications using terrain models (Schneider, 2001b).

Reconstruction of continuous surfaces from a set of points and lines is underdefined (Martinoni, 2002; Wood, 1998). This has two consequences for terrain modelling. First, it is important to use as much of the information as possible included in the input data to reduce the uncertainty of shape (Schneider, 2001a). Not only elevation values at data points, but also the semantic information of peaks, linear features, breaklines, contours, soft breaklines need to be exploited for the surface reconstruction (Figure 1.3). Knowledge about the phenomenon to be modelled is also implicit information. Artifacts of the interpolation method used for reconstructing the surface are suboptimal if they are unplausible, as suggested by the knowledge about the phenomenon. Serious artifacts in the reconstructed surface therefore mean that the implicit information is not well used. An example of such artifacts are the horizontal triangles which occur when interpolating contour data with a linear TIN. According to knowledge about fluvial geomorphology, those horizontal areas are highly unplausible. Depending on the situation it is more likely that they are peaks, valleys, passes or slopes. In order to better use implicit information, a number of strategies have been developed to avoid such artifacts (Brändli, 1991; Heitzinger and Kager, 1998; Thibault and Gold, 2000; Schneider, 1998; Wise, 1997).

Second, application specific requirements and assumptions have to be included into the terrain modelling process (Martinoni, 2002; Schneider, 2001b). For instance, a terrain model used for hydrological modelling in an area where fluvial processes are dominant requires the absence of spurious

sinks. Another example would be a terrain model which is used for visualisation and should have a visually pleasing appearance.

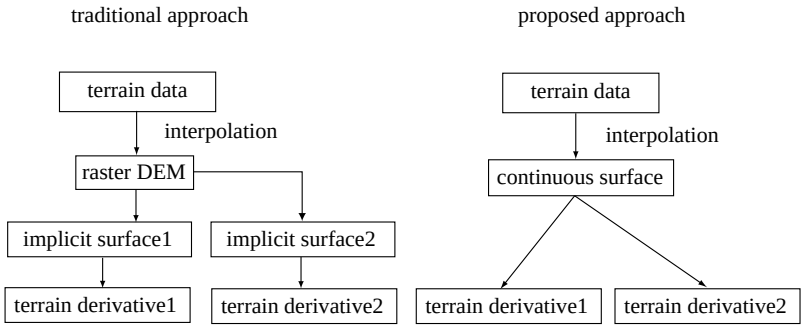

Figure 1.2: Traditional and proposed approach for terrain modelling

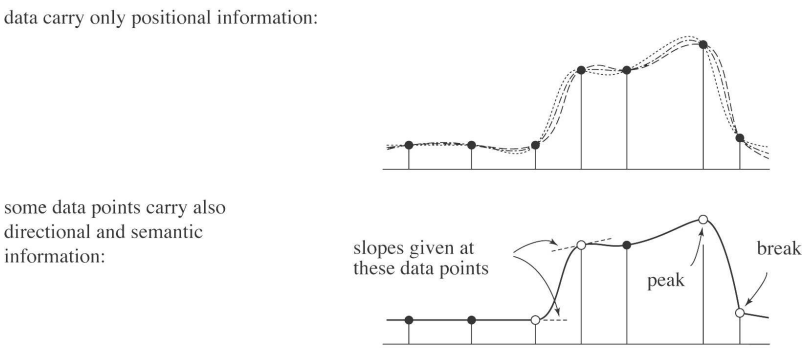

Figure 1.3: More information per datum reduces the uncertainty of local shape [Schneider, 2001a].

A large number of methods exist for the interpolation of terrain surfaces. Two broad classes can be distinguished: point based methods, such as inverse distance weighting, kriging and minimum curvature splines; and methods based on rectangular or triangular tessellations. Although Chapter 2 also reviews point based methods, the focus of this thesis is on triangle based methods.

The tessellation based methods commonly use piecewise polynomial surfaces, such as linear interpolation or bivariate quintic interpolation. A detailed review of methods already applied to terrain modelling can be found in Chapter 2.

Various piecewise methods for surface construction have been developed in the discipline of computer aided graphics design (CAGD) (Beach, 1991; Farin, 1997; Farin et al., 2002; Piegl and Tiller, 1997; Rogers and Adams, 1990). They have been successfully applied to design problems in different domains, e. g. car bodies, aircrafts and ships.

Due to a number of properties, CAGD methods have also been recognised to be highly suitable for the specification of continuous terrain models (Pfeifer, 2002; Schneider, 1998):

- the shape of the surfaces can be controlled locally, thus additional knowledge and application specific requirements can be considered;
- as CAGD methods are tessellation based, linear information (e. g. breaklines) can be incorporated;

- because of their frequent use for design purposes, methods with intuitive shape control exist;
- due to their local, piecewise nature, CAGD methods are in general computationally efficient;

Nevertheless, terrain data and applications are different from the standard applications of CAGD. Therefore, the standard methods will need to be adapted. Breaklines indicating sudden changes in slope are quite common for terrain data, but there is no reference in the CAGD literature known to the author covering this topic. Tessellation artifacts are important in CAGD as well as in terrain modelling. However, in terrain modelling irregular linear input data is more common. Therefore, tessellation elements are more likely to be of long and thin shape. As non-linear methods tend to swing out at such elements, artifacts caused by the tessellation tend to have a stronger impact in terrain modelling.

1.1.4 Choice of methods

There are two broad classes of CAGD methods: methods based on triangles and methods based on rectangles. Triangle based methods can be applied to any data, whereas quadrilateral ones need to have regularly spaced input. This thesis concentrates on triangular methods. The relevance of scattered data interpolation for terrain modelling is high, as most terrain data capture methods produce irregularly spaced output. Moreover, filtered grids are also an application area for scattered data interpolation.

This thesis aims at evaluating and extending the following two methods:

1. The *Triangular Coons patch* is a method to create surfaces interpolating to given boundary curves and boundary derivatives. Therefore, it seems to be an intuitive approach to interpolate to scattered points and to linear features simultaneously. Furthermore, because the boundary curves can be of any type, they provide more flexibility compared to other CAGD methods. As triangular Coons patches have not been used for terrain modelling so far, an evaluation of this technique would be of value.
2. *Triangular Bézier patches* have been well examined due to numerous publications in the CAGD literature. With the Bernstein form, a computationally efficient and numerically stable formulation exists for this class of methods.

1.2 Research questions

The central research question of this thesis is:

- How can Coons patches and triangular Bernstein-Bèzier methods be adapted for continuous terrain modelling?

From this general question, two specific questions arise:

1. How can linear information be included in these methods?
2. What artifacts occur and how can they be avoided?

The evaluation of the generated surfaces will be an important part of this thesis. First, general issues about the comparison of interpolated surfaces will be discussed. Then Coons patches and Bézier splines will be evaluated qualitatively by means of visual inspection and quantitatively using first an artificial surface and, second, measured values of a real terrain surface. Research questions referring to the evaluation of the developed methods are therefore

3. Does the removal of artifacts provide ’better‘ surfaces?

4. Are the proposed methods ’better‘ than the methods already known in terrain modelling?

Last but not least, a software prototype implementing the methods described in this thesis will be developed.

1.3 Thesis outline

Chapter 2 gives an overview of current methods used in digital terrain modelling. Chapter 3 explains the mathematical basics of the triangular Coons patch and Chapter 5 provides an introduction into triangular Clough-Tocher Bézier splines. Chapter 4 explains the consideration of breaklines in Coons patches and chapter 5 does the same for Clough-Tocher Bézier splines. In Chapter 6, the Ruppert algorithm for refinement of triangular meshes is applied to triangular terrain models. A brief description of the prototype software implementing the methods used in this thesis is provided in Chapter 7. Chapter 8 contains an evaluation of the developed method with a field experiment and an artificial surface. The thesis is closed by a general discussion in Chapter 9, as well as conclusions and an outlook in Chapter 10.

Chapter 2

Literature review

2.1 Introduction

In this chapter key terms of digital terrain modelling are defined. Then, a selection of interpolation methods applied to terrain modelling is reviewed:

- *Inverse distance weighting* is perhaps the first approach coming to mind to solve the interpolation problem for scattered point data. Inverse distance weighting, however, is rarely used for terrain interpolation. The reason why it is introduced in this chapter is because kriging and minimum curvature splines are extensions of it.
- *Kriging* was developed in the mining industry but has often been used for the interpolation of various phenomena in GIS, including terrain.
- *Minimum curvature splines* are closely related to kriging. Variations are widely used and implemented within GIS.
- *Linear interpolation* in triangles is the simplest technique for triangle based interpolation.
- *Bivariate quintic interpolation* was the first triangle based method applied to terrain modelling which accounted for G_1-continuous surfaces.
- *Clough-Tocher Bézier splines* provide a method for G_1-continuous surfaces which is only of cubic degree.

Not reviewed are methods which are specifically designed for the interpolation of terrain surfaces from isolines. A review of these can be found in Schneider (1998).

3D views of a test data set (the mountains Zuestoll and Schibenstoll in Switzerland) have been rendered to illustrate the geometric properties of the reviewed interpolation methods. The dataset consists of mass points, structure lines and a breakline, all digitised from the 1:25000 map. Figure 2.1 shows the test data set in an orthogonal projection. Figure 2.3 shows a photograph of Schibenstoll and Zuestoll from the north direction.

2.2 Definition of key terms

G_n-continuity Geometric continuity. Curves and surfaces are G_n-continuous, if the derivatives up to order n vary continuously. G_0-continuity therefore means that the elevation function varies continuously (no gaps). G_1-continuity denotes continuity of slope and G_2-continuity continuity of curvature.

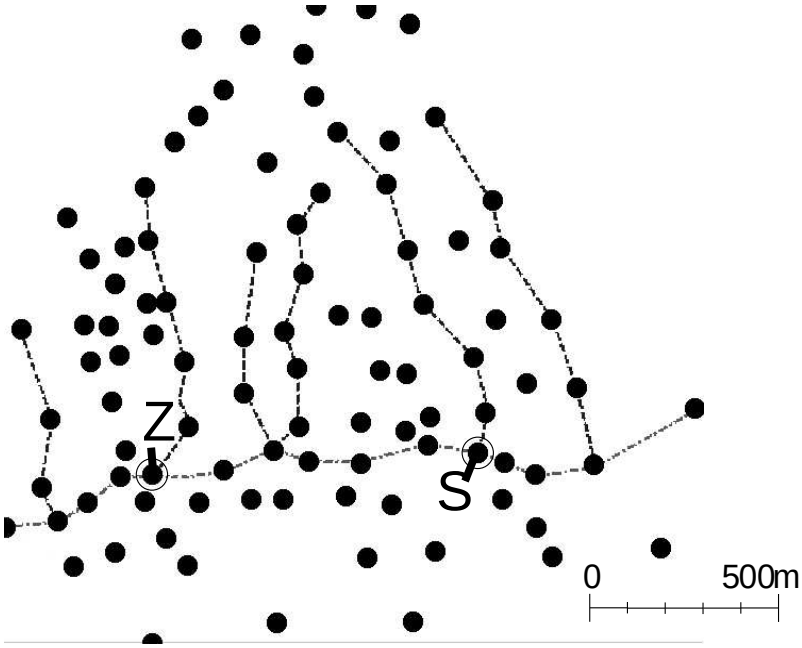

Figure 2.1: Test data set used for the 3D projections. Data points are indicated by black dots, structure lines by dashed lines and breaklines by dash-dot lines. The peaks of Zuestoll (Z) and Schibenstoll (S) are circled

C_n-continuity Parametric continuity. The derivatives of parametric curves and surfaces are vectors, in contrast to normal functions, where the derivatives are scalar. A curve or surface is C_n continuous, if the derivative vectors up to order n vary continuously. Note that, for a curve or surface, the order of C-continuity is not necessarily the same as the order of G-continuity. The order of C-continuity and G-continuity, respectively, will differ if the direction of the derivative vectors varies continuously, but not the length.

Scattered points Arbitrarily spaced and arranged points. Points arranged in a regular grid would, as a special case, also fall into this category.

Breaklines Breaklines (often also termed 'hard breaklines') represent sharp linear terrain features. Across these lines, the slope of the terrain surface is supposed to change suddenly (only G_0-continuity is reached). Sharp ridges are an example of a situation where breaklines may be used (Figure 2.2).

Structure lines Linear terrain features (also called 'soft breaklines' by some authors). The difference from breaklines is that the slope is not supposed to change suddenly (G_1-continuity). Usually, geomorphologic features like valleys and obtuse ridges are modelled as structure lines (Figure 2.3).

Mass points In a dataset where scattered data points and linear elements are present, the term mass point is commonly used to denote a scattered data point, to make a clear distinction to the vertices of the linear elements.

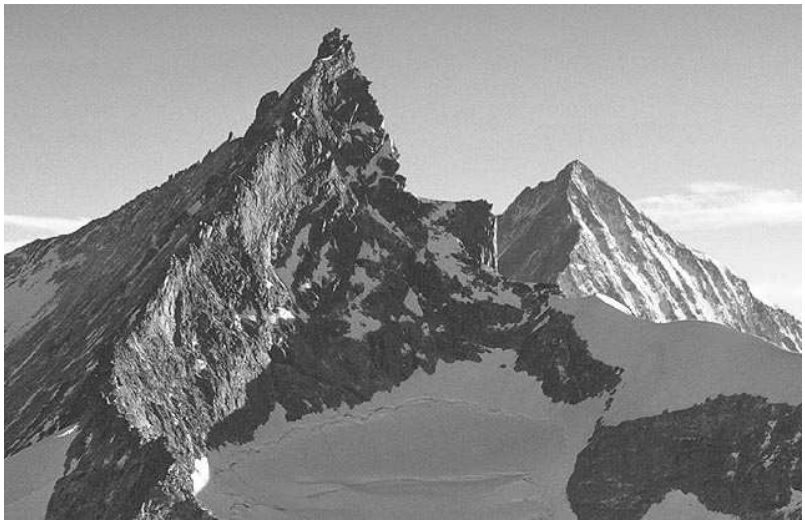

Figure 2.2: Zinalrothorn (Wallis, Switzerland). The sharp ridge in the middle of the picture is a nice example of a geomorphologic feature where a breakline would typically be inserted.

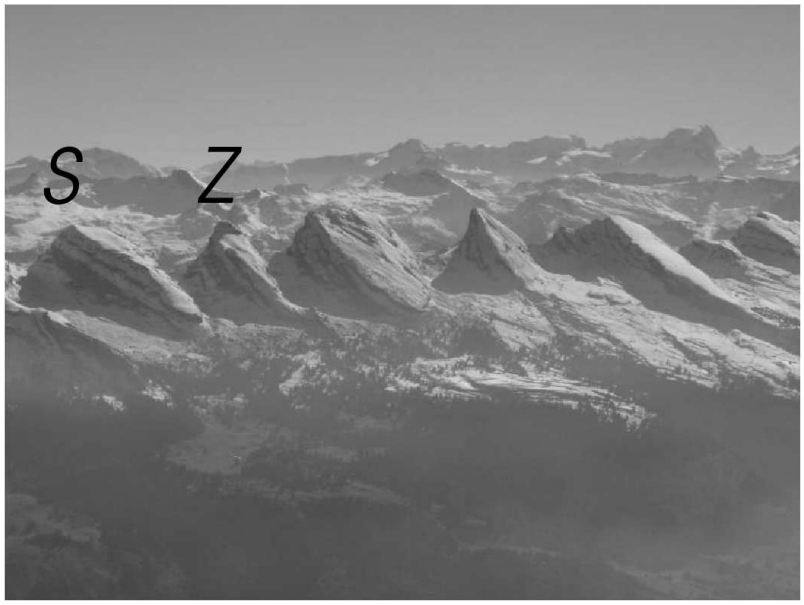

Figure 2.3: Churfirsten (Switzerland). The obtuse ridges facing towards the viewer are nice examples of geomorphologic features where structure lines would typically be inserted. The positions of the mountains Zuestoll and Schibenstoll, which are used as a study area in this thesis, are marked.

Interpolation, extrapolation Interpolation is the procedure of predicting the value of attributes, elevation for digital terrain models, at unsampled sites from measurements made at locations within the same area or region. Prediction of values outside the area covered by existing observations (typically designated by the convex hull of the data points) is called extrapolation (Burrough and McDonnell, 1998).

Interpolation point, data point For the sake of clarity, in this thesis, an unsampled location where a value needs to be interpolated is called *interpolation point*. Sample points or vertices of sample lines are referred to as *data points*.

Delaunay triangulation Triangulation which has been built using the Delaunay criterion. This criterion maximises the minimum angle and thus accounts for relatively compact triangles. A property often used by construction algorithms is that the circumcenter of each triangle does not contain any other data point (Fortune, 1995; Heller, 1990). To include lines which are not supposed to cross a triangle edge the empty circle property can be relaxed across these lines, resulting in a so-called *constrained Delaunay triangulation* (de Floriani and Puppo, 1992). In digital terrain modelling, constrained lines are often used to make sure that valleys and ridges are properly represented.

Voronoi polygons, Voronoi diagram In a point dataset, the Voronoi polygon of a point is the area, which is closer to this point than to any other data point. All Voronoi polygons of a point set together form the (point) Voronoi diagram. The Delaunay triangulation and the Voronoi diagram are geometric duals. The Delaunay triangulation of a dataset can be obtained by connecting points, which belong to adjacent Voronoi polygons (Fortune, 1995).

Proximal interpolation The elevation of an interpolation point is the elevation of the closest data point. On a surface interpolated with proximal interpolation, all points within a Voronoi polygon have the same value (Watson, 1992).

Interpolation vs. approximation Interpolation methods provide curves and surfaces which pass exactly through the data points. Approximation methods, by contrast, yield curves and surfaces which do not necessarily have to pass through the data points.

2.3 Inverse distance weighting

2.3.1 Method

The basic idea of the inverse distance method is the *First Law of Geography*: "Things which are closer to each other are more related than distant ones" (Tobler, 1970). Therefore, the unknown elevation of an interpolation point is calculated by weighting the elevations of the data points according to their distance and averaging them. Given a set $P = P_1, \ldots, P_N$ containing N data points, the elevation z of an interpolation point (x,y) would then be calculated using the following formula:

$$z = \begin{cases} z_i & x = x_i \wedge y = y_i \\ \frac{\sum_{i=1}^{N} W_i * z_i}{\sum_{i=1}^{N} W_i} & \begin{pmatrix} x \\ y \end{pmatrix} \notin P \end{cases} \tag{2.1}$$

with W_i (the weight of point i) being a function which decreases with distance. A common choice for W_i is $\frac{1}{d_i{}^p}$, where d_i is the distance between the interpolation point (x/y) and a data point i. p is a parameter describing how fast the weight of a data point decreases with distance (Watson, 1992).

The choice of p has a considerable influence on the shape of the resulting surface. If p is small, the weight of a data point decreases rapidly with distance. For $p < 1$, the surface forms spikes at the data points and is not differentiable. The surface approaches the mean of all elevations the closer p approaches 0. If $p > 1$, the surface is horizontal at the data points. The surface approaches a proximal

interpolation the higher p gets. Figures 2.4 - 2.8 show the effect of changing the value of p in the 2-dimensional case. Figures 2.9 - 2.12 show results of a 3-dimensional terrain surface.

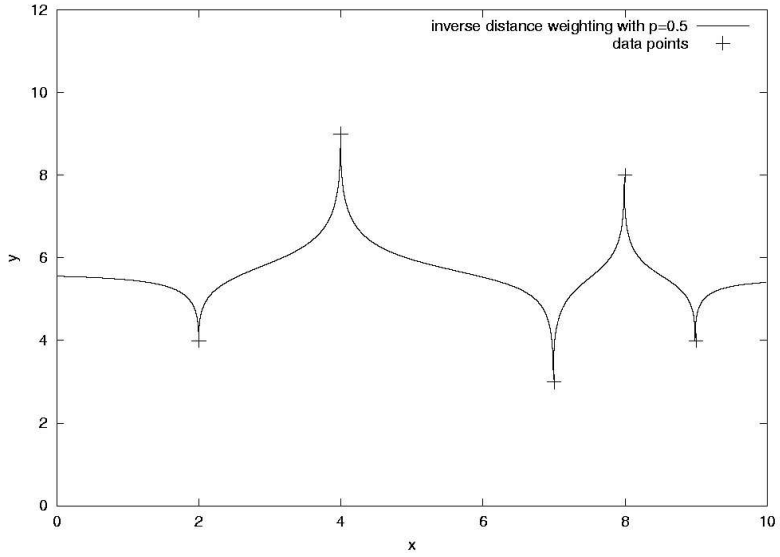

Figure 2.4: Inverse distance weighting with $p = 0.5$

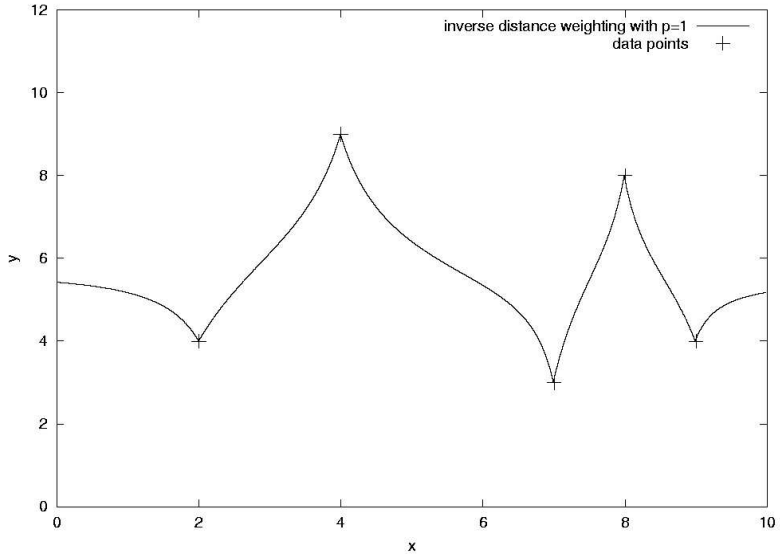

Figure 2.5: Inverse distance weighting with $p = 1$

2.3.2 Discussion

Inverse distance weighting is intuitive and can be used even if the distribution of the data points is such that methods based on a tessellation cannot be applied (e. g. data points on a single profile).

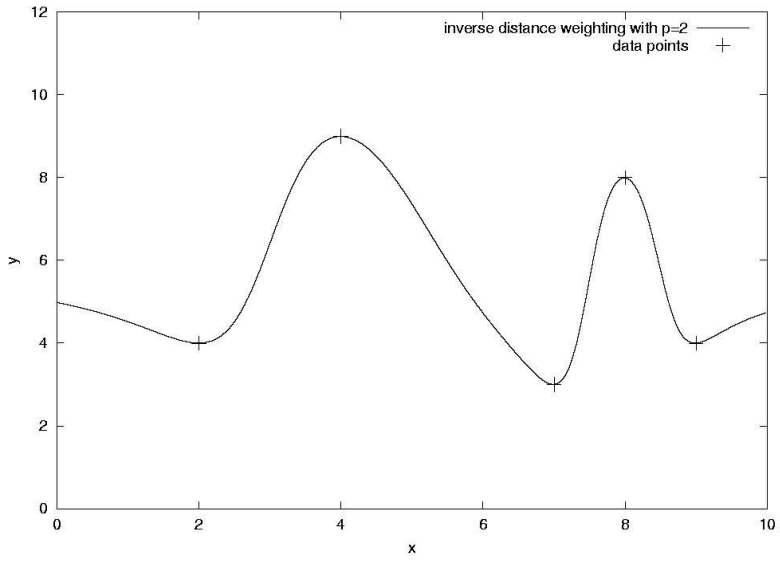

Figure 2.6: Inverse distance weighting with $p = 2$

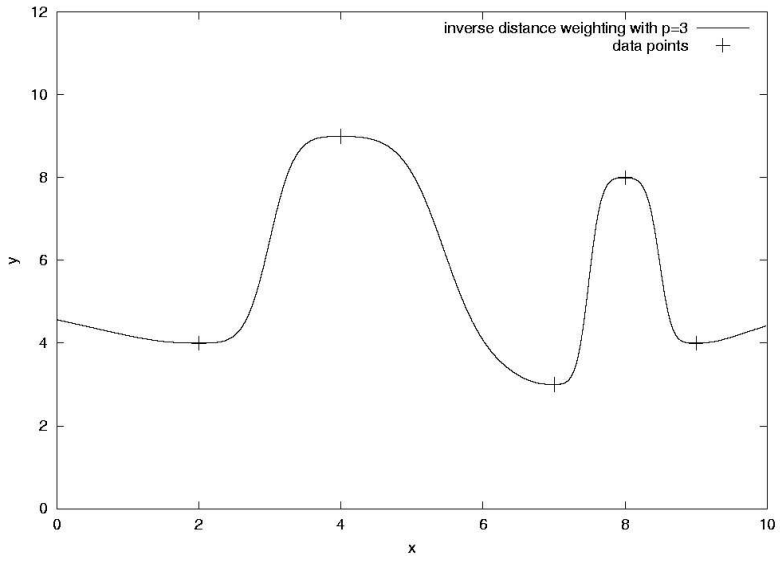

Figure 2.7: Inverse distance weighting with $p = 3$

However, there are numerous disadvantages (Watson, 1992):

- As figures 2.4 - 2.8 show, the choice of the parameter p is crucial. A suitable value of p has to be estimated by the user by means of experiment. Different strategies are possible for this. Surfaces with different values for p can be calculated and a suitable value for p can be chosen by rendering the surfaces and comparing them visually. Alternatively, a cross-validation approach can be used. For this, surfaces with different values for p are constructed using only a subset of the available data. The remaining data points are used for validation and the p with

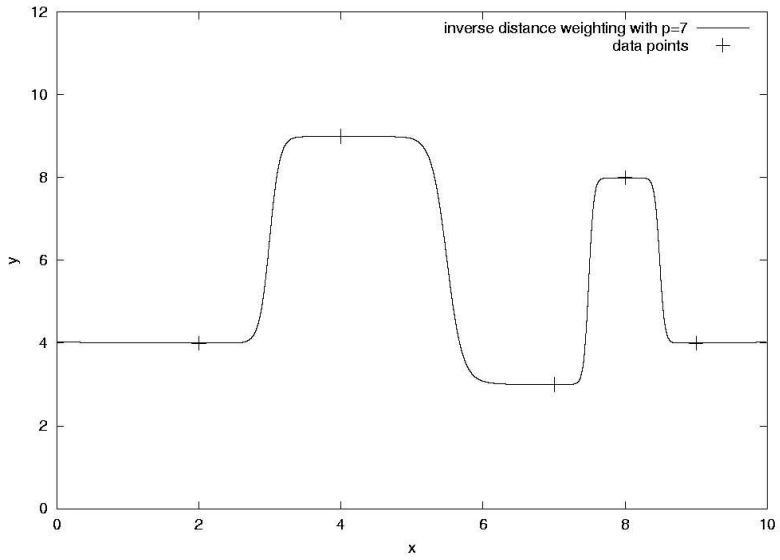

Figure 2.8: Inverse distance weighting with $p = 7$

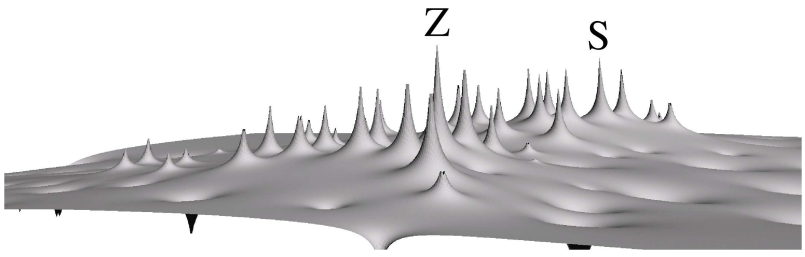

Figure 2.9: Inverse distance weighting with $p = 1$

the lowest sum of deviations between the validation points and the corresponding surface is selected. However, both approaches are time consuming and highly empirical.

- The values of the surface are between the lowest and the highest value given in the data set. Thus, unsampled peaks higher than the maximum elevation cannot be reconstructed.
- The surface can be influenced between the two extreme cases mean value ($p \to 0$) and proximal interpolation ($p \to \infty$). Because of this property, unsampled peaks within the maximal value can also not be reconstructed. Figure 2.13 shows a situation with an unsampled peak. As inverse distance weighting is not capable of reconstructing this implicitly contained feature, it does not make use of all available information for the specification of a continuous terrain surface.

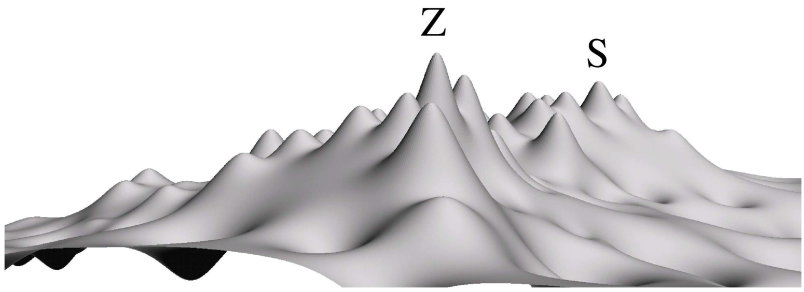

Figure 2.10: Inverse distance weighting with $p = 2$

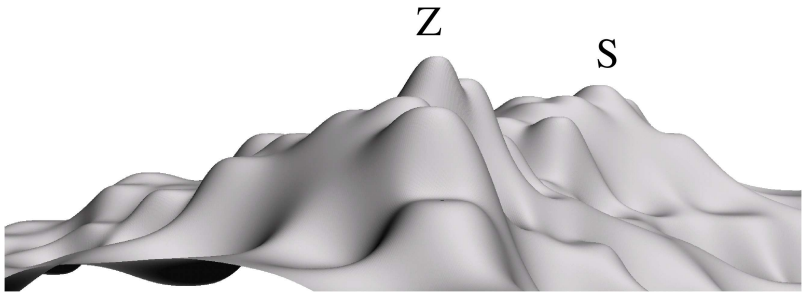

Figure 2.11: Inverse distance weighting with $p = 3$

- The influence of a data point is the same in all directions. The method produces radially symmetric peaks or pits at each data point. Because of this effect, linear features like valleys or ridges are obscured (Watson, 1992). Lines given as part of the input data cannot be enforced because of this isotropy.

- As inverse distance weighting is a global method, the computation costs may be high using large datasets. A common extension, which overcomes this drawback, is to consider only data points within a certain distance about point (x,y), essentially turning inverse distance weighting into a local method. With this approach, however, the resulting surface is not G_0-continuous any more.

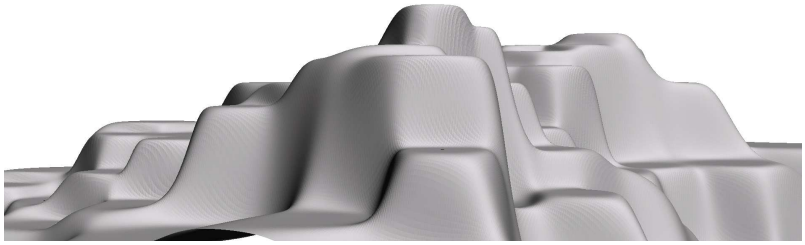

Figure 2.12: Inverse distance weighting with $p = 7$

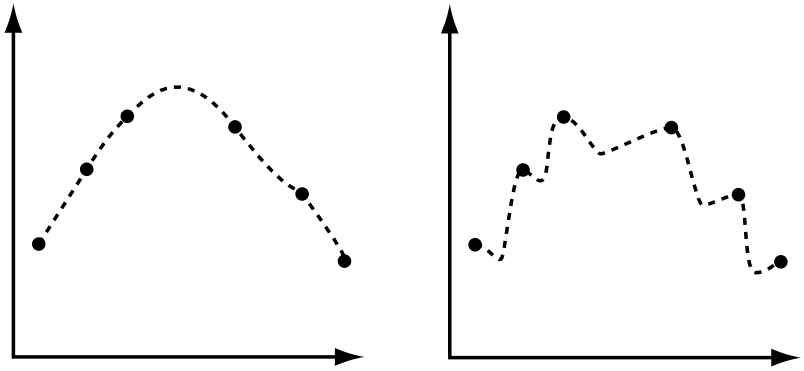

Figure 2.13: Interpolation of a dataset containing an unsampled peak.

2.3.3 Inverse distance weighted gradients

Inverse distance weighted gradients is a modification to inverse distance weighting which overcomes some of the deficiencies described in 2.3.2. First, normals at the data points have to be estimated. Then, inverse distance weighting is applied, but instead of the elevation values at the data points, the elevations of the tangent planes at the location of the interpolation point are weighted and averaged. This modification allows the reconstruction of unsampled peaks and pits and avoids the stepped, terraced appearance of the surface (Watson, 1992).

2.4 Kriging

For inverse distance weighting, the weights given to the data points have to be estimated (by the choice of the parameter p). Kriging is a method to choose these weights based upon geostatistical analysis.

It has been developed by Krige and Matheron for use in the mining industry (Matheron, 1962, 1963).

For the application of kriging, the spatial variation of a variable is divided into three components (Figure 2.14) (Wackernagel, 1998):

- a global trend. This trend function is usually simple (a constant mean or a function of low degree)
- a random but spatially correlated variation
- a spatially uncorrelated variation (noise).

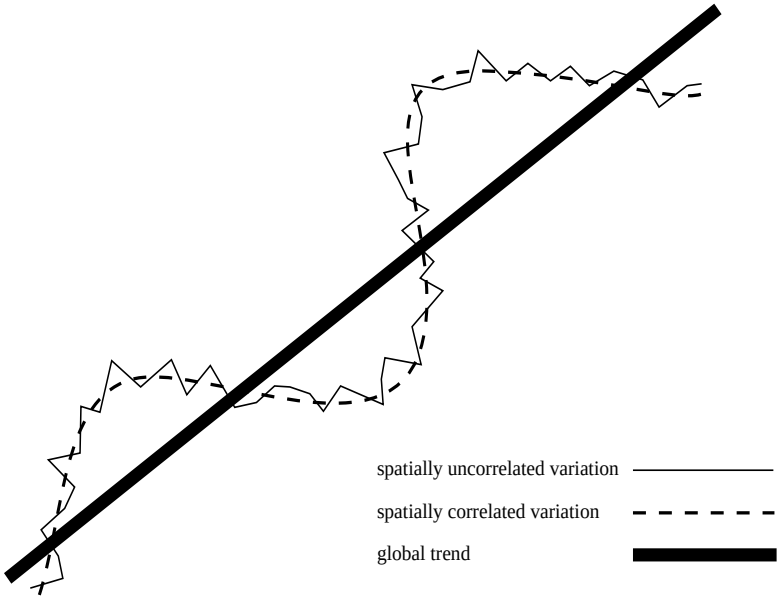

Figure 2.14: Spatial variation of a variable

2.4.1 Semivariogram

While the trend is usually modelled with a constant value or a function of low degree, the second, spatially correlated component can be modelled with a *semivariogram*. The semivariance between several data points (in practice usually two), separated by distance h is defined as

$$\frac{1}{2n}\sum_{i=1}^{n}(z(x_i)-z(x_i+h))^2. \tag{2.2}$$

Note that the z-values in this equation are the deviations of the data points from the trend surface and not the elevation values of the data points. The semivariogram is a plot of the semivariance between pairs of data points against the distance between them. Figure 2.15 shows an example of such a plot, together with a fitted function, showing three characteristic parameters of a variogram (Isaaks and Srivastava, 1989):

- The *range* is the distance at which the variogram reaches its plateau. Below this distance, the semivariance increases with growing distance between the points. For the estimation of the

random and spatially correlated variation this means that only data points within the range have to be taken into account.

- The semivariance value at which the variogram levels off is called *sill*.
- The value of the fitted function for $h = 0$ is called the *nugget*.

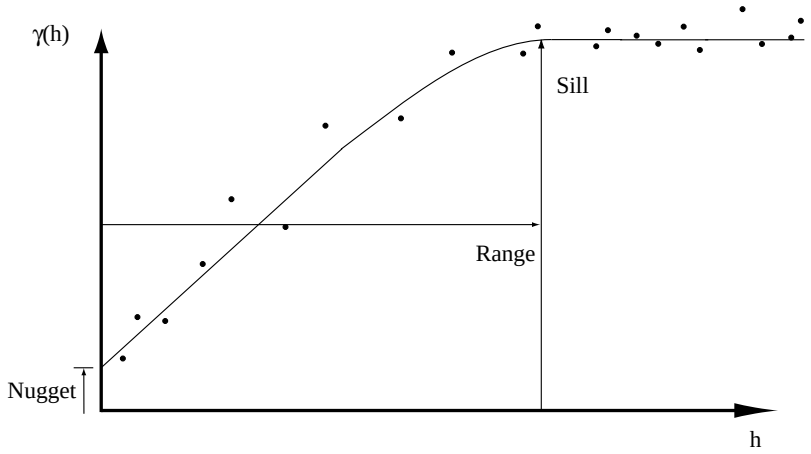

Figure 2.15: Empirical variogram fitted using a spherical model. $\gamma(h)$ is the semivariance between points and h the lag (distance between points)

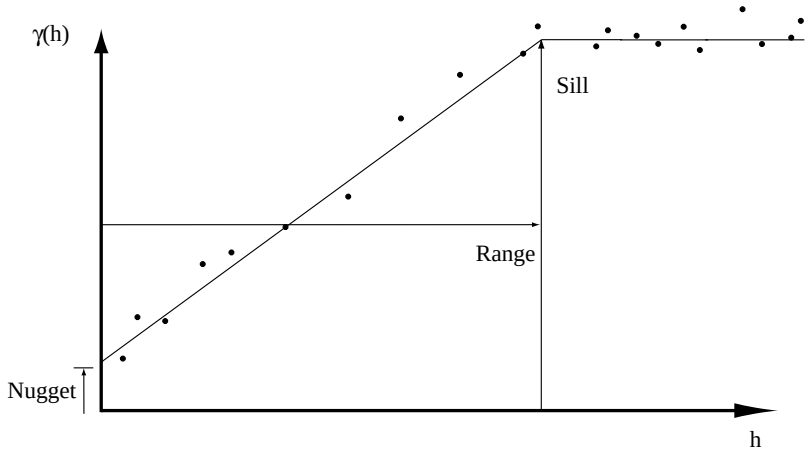

Figure 2.16: Empirical variogram fitted using a linear model. $\gamma(h)$ is the semivariance between points and h the lag (distance between points)

If estimates for the nugget (c_0), the sill ($c_0 + c_1$) and the range (a) are available, different types of functions can be used to fit the points of a variogram. The most common ones are (Isaaks and Srivastava, 1989):

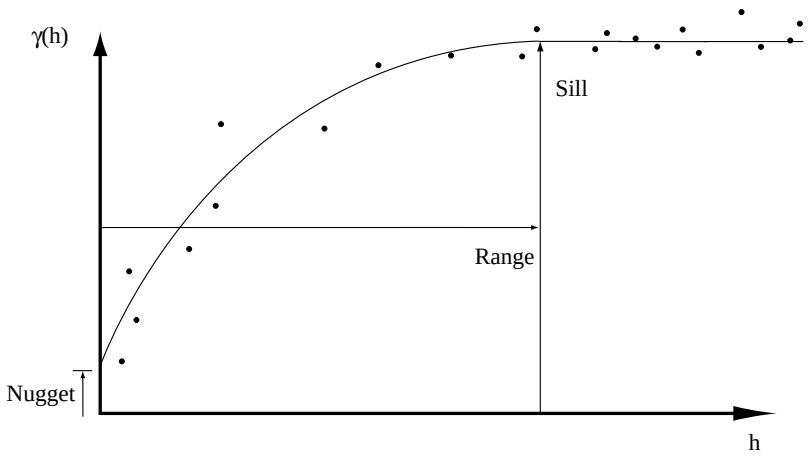

Figure 2.17: Empirical variogram fitted using an exponential model. $\gamma(h)$ is the semivariance between points and h the lag (distance between points)

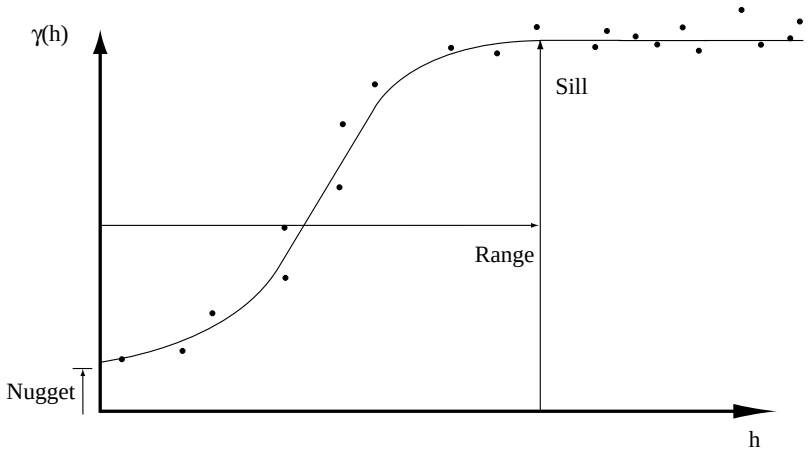

Figure 2.18: Empirical variogram fitted using a Gaussian model. $\gamma(h)$ is the semivariance between points and h the lag (distance between points)

- the spherical model (figure 2.15)

$$\gamma(h) = \begin{cases} c_0 + c_1 \cdot \left(\frac{3h}{2a} - \frac{1}{2}\left(\frac{h}{a}\right)^3\right) & 0 \leq h \leq a \\ c_0 + c_1 & h > a \end{cases} \tag{2.3}$$

- the linear model (figure 2.16)

$$\gamma(h) = c_0 + bh, \tag{2.4}$$

  where b is the slope

- the exponential model (figure 2.17)

$$\gamma(h) = c_0 + c_1 \cdot \left(1 - exp(\frac{-h}{a})\right) \tag{2.5}$$

- the Gaussian model (figure 2.18)

$$\gamma(h) = c_0 + c_1 \cdot \left(1 - exp(\frac{-h}{a})^2\right) \tag{2.6}$$

A variogram which can be fitted by a Gaussian model is typical for smoothly varying data. This situation often occurs with elevation data (Burrough and McDonnell, 1998).

Fitting variograms is usually done interactively because of the nonlinearity of common variogram models and the number of parameters to estimate (Chiles and Delfiner, 1999). This can be accomplished by selecting a model and by specifying a subset of parameters of this model. The remaining parameters can then be calculated automatically with a fitting method, e. g. a least squares fit (Pebesma, 2000). Knowledge about the interpolated phenomenon can be included into the specification of the variogram. One example is the choice of the behaviour of a variogram function near the origin. If the data contains no measurement error, a function with no or with a very small nugget can be chosen. If, in contrast, the phenomenon under study is known to have a very regular spatial structure (e. g. gravity or magnetism) and it is assumed that there are measurement errors, a function with a stronger nugget can be chosen, e. g. by using a spherical model. User knowledge is further needed to consider various special cases in variogram modeling. If, for instance, the nugget is dominant compared to the autocorrelated component, kriging interpolation does not make any sense and the best estimation for the variable is the trend (Chiles and Delfiner, 1999).

After the variogram fitting, the following steps are carried out using the fitted function (model semivariogram). The semivariances between the pairs of points (empirical semivariogram) are not used further.

Four approaches to kriging are now introduced: ordinary kriging, block kriging, universal kriging and stratified kriging.

2.4.2 Ordinary Kriging

In ordinary kriging the value z of a spatially varying variable at an interpolation point can be calculated using the weighted sum of the values at the n data points within the range:

$$z = \sum_{j=1}^{n} w_j * z_j. \tag{2.7}$$

The estimated semivariance function can now be used to determine the weights w_j.

Ordinary kriging aims to minimise the error variance $\sigma_R^2 = \frac{1}{k}\sum_{i=1}^{k}(r_i - m_R)^2$, where r_i is the error of an estimate and m_R the mean error of k estimates. As the r_i as well as m_R are unknown, it is not possible to minimise the variance of errors. The approach of kriging is to build a model of the data and to work with the mean error and error variance of this model (Isaaks and Srivastava, 1989). The minimum error variance of the kriging model is obtained when

$$\begin{bmatrix} \gamma(x_1,x_1) & \cdots & \gamma(x_1,x_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(x_n,x_1) & \cdots & \gamma(x_n,x_n) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ \phi \end{bmatrix} = \begin{bmatrix} \gamma(x_1,x_0) \\ \vdots \\ \gamma(x_n,x_0) \\ 1 \end{bmatrix}, \tag{2.8}$$

or, in matrix notation,

$$C \cdot w = D, \tag{2.9}$$

where $\gamma(x_i, x_j)$ denotes the semivariance between the data points i and j, which is provided, as a function of the distance between the two points, by the variogram. $\gamma(x_i, x_0)$ denotes the semivariance between the data point i and the interpolation point. ϕ represents the Lagrange parameter needed for minimisation (Chiles and Delfiner, 1999). The vector containing the weights and the Lagrange parameter therefore can be calculated using matrix inversion:

$$w = C^{-1} \cdot D. \tag{2.10}$$

Ordinary kriging is an exact interpolator. If x_0 is equal to a point of $x_1, \cdots, x_n$, the only solution of the equation system is to set the weight of this data point to one and all the others to zero.

As the kriging variance can be calculated at each interpolation point, it can be used as a measure for the quality of an interpolated value. Isaaks and Srivastava (1989) provides an excellent explanation of the details.

Figure 2.19 shows the test dataset interpolated with ordinary kriging. The perspective plot reveals spikes and pits at the data points and a smooth surface otherwise. These artifacts are similar to the ones occurring in inverse distance interpolation and show the close relationship between the two methods.

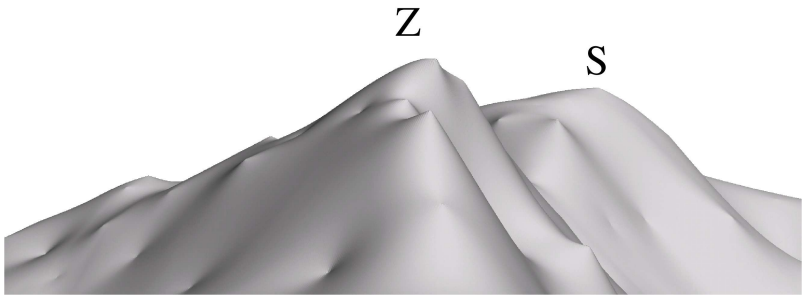

Figure 2.19: Data set Zuestoll interpolated with ordinary kriging. The surface passes through the data points, but shows spikes and pits at these locations. The variogram has been estimated using the mean as trend function and a spherical function with range 400m for variogram fi tting.

2.4.3 Block Kriging

Kriging can be modified to calculate an average value over blocks. Block kriging provides a way to calculate such averages by solving only one kriging equation system. This is done by replacing the vector D in equation 2.9 by a Vector containing the semivariances between each data point and the block B to be estimated (Burrough and McDonnell, 1998):

$$\begin{bmatrix} \gamma(x_1,x_1) & \cdots & \gamma(x_1,x_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(x_n,x_1) & \cdots & \gamma(x_n,x_n) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ \phi \end{bmatrix} = \begin{bmatrix} \gamma(x_1,B) \\ \vdots \\ \gamma(x_n,B) \\ 1 \end{bmatrix}. \tag{2.11}$$

The semivariance between a point and a block is calculated by discretising the area of the block into several points and by averaging the semivariances between the point and the points of the block (Isaaks and Srivastava, 1989).

Block kriging does not have the spikes and pits which occur in ordinary kriging and universal kriging, but the surface does no longer pass through the data points. Figure 2.20 shows a perspective view of the test data set interpolated using block kriging with a block size 40 m*40 m.

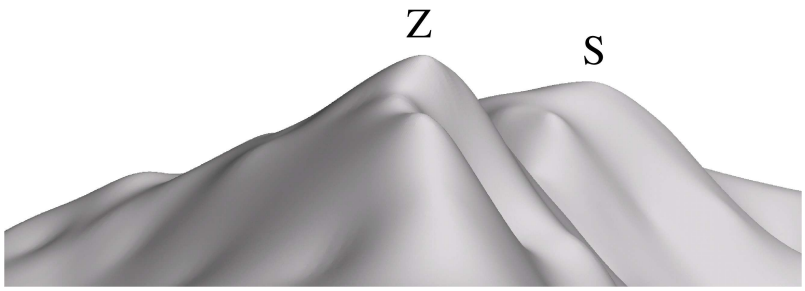

Figure 2.20: Block kriging with a block size of 40m * 40m. The variogram has been estimated using the mean as trend function and a spherical function with range 400 m for variogram fi tting.

2.4.4 Universal kriging

Universal kriging incorporates the trend into the kriging functions. For a first order trend function the parameters b_0, b_1 and b_2 of the function $z = b_0 + b_1 x + b_2 y$ have to be estimated. Therefore the following equation system can be used to estimate the coefficients b_n and the weights (Watson, 1992):

$$\begin{bmatrix} 1 & x_1 & y_1 & \gamma(x_1,x_1) & \cdots & \gamma(x_1,x_n) \\ \vdots & \ddots & & \vdots & \ddots & \\ 1 & x_n & y_n & \gamma(x_n,x_1) & \cdots & \gamma(x_n,x_n) \\ 0 & 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & x_1 & \cdots & x_n \\ 0 & 0 & 0 & y_1 & \cdots & y_n \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \gamma(x_1,x_0) \\ \vdots \\ \gamma(x_n,x_0) \\ 1 \\ x \\ y \end{bmatrix} \tag{2.12}$$

2.4.5 Stratifi ed kriging

An extension to kriging classifies the area into subareas and uses semivariograms for each subarea. The interpolation is then carried out separately for each subarea (Burrough and McDonnell, 1998). A similar extension is to use polylines or polygons as boundaries and to only use points on the same side as the requested points for the weighted average (Pebesma, 2000). Unfortunately, in both methods the surface is not G_0-continuous at the boundaries of the subareas.

2.4.6 Discussion

A major advantage of kriging is that, in contrast to inverse distance weighting, the range and the weights for interpolation are derived from a geostatistical analysis. Furthermore, the kriging estimation variance is available and provides a measure for the quality of each interpolated value. Kriging is often referred to as an 'optimal interpolator' (Burrough and McDonnell, 1998). However, it is optimal only if the assumptions made for the chosen version of kriging as well as the model of autocorrelation estimated with the variogram are appropriate. The same holds for the estimation variance as a measure for the quality of interpolation.

Fitting of variograms is an interactive process requiring considerable skills from a user (Burrough and McDonnell, 1998). Thus, kriging is not an 'easy to use' method. Furthermore, some disadvantages of inverse distance weighting remain: the method is point based; linear features (which are far more important in terrain modeling than in other applications) can only be considered by introducing discontinuities. The artifacts in ordinary kriging (spikes and pits) can be removed using block kriging. However, a resulting property of the surface is that it does not pass through the data points and local shapes may no longer be represented.

Kriging is often used in the geosciences. Erxleben et al. (2002) used kriging to interpolate snow depth. Jimenez-Espinosa and Chica-Olmo (1999) interpolated gold contents found in soil samples in Spain. Martinez-Cob (1996) interpolated evapotranspiration and precipitation using kriging. There are several reasons for the frequent usage of kriging interpolation. Due to historical reasons, kriging is often used for mining and geochemical applications. It has also been applied for the interpolation of ice sheet surfaces, where widespread previous use have established its use (Herzfeld, 1999; Fricker et al., 2000). An important property for many applications is the possibility to include other variables, on which the interpolated phenomenon is dependent, into the interpolation model (cokriging). Temperature values, for instance, are influenced by elevation and interpolation of temperature thus can be improved by considering elevation.

Kriging has not often been applied for interpolation of terrain surfaces. An exception is Gao (1997), unfortunately without any information as to why kriging is chosen as an interpolation method. For the interpolation of elevation data it is hard to use dependencies between elevation and other variables (e. g. geology) to improve the interpolation. Therefore cokriging is not useful in interpolating terrain surfaces. Terrain data is also special in that it often contains linear elements (contours, breaklines, structure lines) besides the point data. Therefore, the limited possibility to include such lines in the surface in kriging is more important in terrain modeling than in other application domains which may exclusively use point data for surface reconstruction. Furthermore, the separation between trend, autocorrelated variation and uncorrelated noise is hard to achieve for terrain surfaces. Terrain modeling is more a very local and geometric operation than a geostatistical and regional one.

2.5 Minimum curvature spline

A method closely related to universal kriging is the minimum curvature spline (Dubrule, 1984; Watson, 1992). Similar to universal kriging (section 2.4.4), the formula to calculate a value z at an interpolation point (x/y) contains a trend function and a weighted sum of the basis functions $C(P_n - X)$ between each data point and the interpolation point X:

$$z = b_0 + b_1 x + b_2 y + a_1 C(P_1 - X) + \cdots + a_n C(P_n - X). \tag{2.13}$$

In the case of minimum curvature splines, the basis function $C(P_1 - P_2)$ equals $d^2 \log d$, where d is the distance between P_1 and P_2 Watson (1992). The weights $a_1, \cdots, a_n$ as well as b_0, b_1 and b_2 can be

calculated by solving the following system of equations:

$$\begin{bmatrix} 1 & x_1 & y_1 & 0 & C(P_1-P_2 & \cdots & C(P_1-P_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & C(P_n-P_1) & \cdots & C(P_n-P_2) & 0 \\ 0 & 0 & 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & x_1 & x_2 & \cdots & x_n \\ 0 & 0 & 0 & y_1 & y_2 & \cdots & y_n \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.14}$$

Mitas and Mitasova (1999) describe an extension with a different basis function, called *Regular spline with tension*. The properties of this method can be adjusted by two parameters: one tunes the tension of the surface and the other tunes the property to pass close to the data points. This method is implemented in Arc/Info (ESRI, 2002) and in Grass (GRASS, 2004).

2.5.1 Discussion

The main difference between minimum curvature splines compared to kriging is that no geostatistical analysis are necessary prior to interpolation. They are preferable if the resulting surface should be smooth, while kriging emphasises the examination of the spatial behaviour of a variable. The advantages and disadvantages of minimum curvature splines and regular splines with tension are similar to those discussed for kriging. An important difference from kriging is that they are global methods. Therefore, the computational cost of these methods is high, and subdivision approaches have been developed to handle this problem (Mitas and Mitasova, 1999).

2.6 Finite Elements

The Finite Element method has been applied in engineering for several decades. Ebner and Reiss (1978) applied this method to calculate grids from scattered points and breaklines. It was, for example, implemented in the computer program HIFI (Ebner et al., 1980).

The principle of the method is to choose the elevation of the grid points, such that the weighted sum of the approximated curvatures and the deviations of the surface from the given data points is minimal. The curvature at a grid point is approximated in *x*- and *y*-direction using the following formulae:

$$z_{xx,i,j} = z_{i-1,j} - 2z_{i,j} + z_{i+1,j} \tag{2.15}$$

$$z_{yy,i,j} = z_{i,j-1} - 2z_{i,j} + z_{i,j+1}, \tag{2.16}$$

where $z_{i,j}$ is the elevation value of $p_{i,j}$.

Figure 2.21 shows how the indexes of points are arranged in a grid. Using the bilinear method, the deviation of the surface from a data point k is given by

$$v_k = (1-\Delta x_k)(1-\Delta y_k)z_{i,j} + \Delta x_k(1-\Delta y_k)z_{i+1,j} + (1-\Delta x_k)\Delta y_k z_{i,j+1} + \Delta x_k \Delta y_k z_{i+1,j+1} - z_k. \tag{2.17}$$

The weighted sum of the z_{xx}, z_{yy} and v_k in a $m \times n$ grid and with s data points can then be minimised:

$$\sum_{k=1}^{s} v_k^2 p_k + \sum_{i=2}^{m-1}\sum_{j=1}^{n} z_{xx,i,j}^2 + \sum_{i=1}^{m}\sum_{j=2}^{n-1} z_{yy,i,j}^2 = min, \tag{2.18}$$

where p_k denotes the weights, which can be specified for each data point (Ebner, 1983). Thus, there is a tradeoff between accuracy (in the sense that the surface passes close to the data points) and smoothness.

With these bilinear elements, the global surface is G_0-continuous. Instead of the bilinear version, a bicubic version is also implemented in HIFI, which makes the global surface G_1-continuous (Ebner, 1983).

Breaklines can be inserted by detecting the intersection points between the breaklines and the grid cells and changing equations 2.15 and 2.16 for all affected grid points, such that there is no connection between consecutive grid points anymore. Breakline support was implemented in HIFI only for the bilinear version, because it would be very complex for the bicubic case (Ebner, 1983).

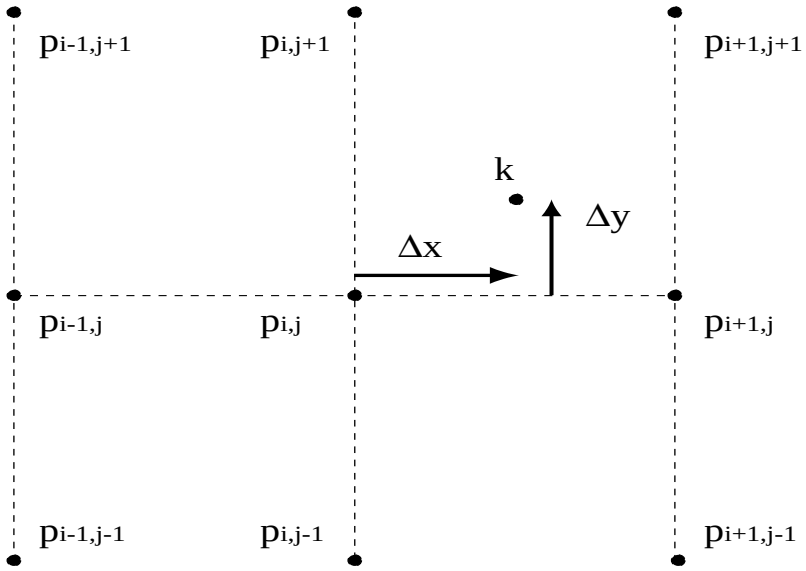

Figure 2.21: Numbering of grid points in the finite element method.

2.6.1 Discussion

In contrast to the previously described methods, the finite element approach introduced by Ebner and Reiss (1978) allows for the integration of breaklines into the surfaces. Therefore, linear structures like ridges and valleys, which are characteristic for landscapes with dominant fluvial or glacial processes, can be represented.

There is a tradeoff, however, between the smoothness of a surface and the accuracy of interpolation. The possibility to choose weights for the data points allows the user to select the option which suits their needs best.

The goal of the described method is clearly the creation of grids. For the specification of continuous terrain models, the direct way would be to use the original data points for linear or cubic interpolation instead of calculating grid points first. As each interpolation introduces uncertainties, auxiliary transformations of the data should be avoided.

There are further disadvantages of this method. Curvature is not really minimised properly, only an approximation in two directions at the grid points is used. The choice of an appropriate cell size is crucial and its determination requires expert knowledge and/or tests using several cell sizes. The

inclusion of breaklines is complicated in the bicubic case (and no implementation is known to the author), while the use of structure lines is not possible for the bilinear as well as for the bicubic case.

2.7 Triangle based methods

A triangulated irregular network (TIN) is a subdivision of an area into triangular facets based on a set of scattered points (Peucker et al., 1978). The principle of triangle based interpolation methods is to create a TIN from a set of data points and to interpolate the surface in each triangle.

Many possibilities exist to build a TIN from a given set of points. In most applications, such as in terrain modeling, it is important to have a triangular mesh with triangles that are as equilateral as possible. To build such a triangulation the Delaunay criterion is often used. Delaunay triangulations have the property that there are no other data points inside the circumcircle of every triangle. As a Delaunay triangulation maximises the minimum angle of a triangulation the triangles are relatively compact. Further details as well as a set of algorithms to build Delaunay triangulations can be found in Fortune (1995).

To represent linear terrain features, such as ridges or valleys, it is necessary to violate the Delaunay criterion at certain locations such that no triangle edge crosses a linear feature. This can be done by a so-called constrained Delaunay triangulation. Definitions and an algorithm for constrained Delaunay triangulation can be found in de Floriani and Puppo (1992).

2.7.1 Advantages and disadvantages of triangle based methods

Interpolation methods based on triangular tessellations have several advantages:

- Compared with methods based on gridded tessellations, the abilities to use irregularly spaced data and to vary the sampling density are important differences. Most terrain datasources are irregularly spaced. Furthermore, these properties are sometimes also an advantage in terms of modeling. Tucker et al. (2001), for instance, use triangle based surfaces for erosion modeling because the horizontal component of erosion processes (e. g. stream meandering) can be modelled more flexibly if the data points can be irregularly spaced.

- The consideration of linear terrain features, such as ridges and valleys, is straightforward using a constrained Delaunay triangulation.

- The computational costs are generally small since triangle based interpolation methods are local.

The triangle based approach also has disadvantages:

- If a triangle edge crosses linear terrain features, such as ridges and valleys, artifacts may be introduced to the surface.

- If there are a lot of constrained edges in a TIN (e. g. when interpolating contour data), the constrained Delaunay triangulation in general has a lot of long and thin triangles (Schneider, 1998), which commonly introduce artifacts.

2.7.2 Linearly interpolated TIN

Once a TIN has been created, the elevation of an interpolation point can be calculated by finding the three vertices of the triangle which contains the interpolation point and interpolating between them. For linear interpolation, the equation of a plane passing through the three vertices can be used. Let (x,y,z) be an interpolation point and we want to find out z, the elevation of this point. Let $P_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$, $P_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}$ and $P_3 = \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix}$ be the three vertices of the triangle containing P. z can then be calculated using the following formula:

$$z = ax + by + c, \tag{2.19}$$

with

$$a = \frac{z_1(y_2 - y_3) + z_2(y_3 - y_1) + z_3(y_1 - y_2)}{(x_1 - x_2)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_2)}, \tag{2.20}$$

$$b = \frac{z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)}{(y_1 - y_2)(x_2 - x_3) - (y_2 - y_3)(x_1 - x_2)}, \tag{2.21}$$

$$c = z_1 - ax_1 - by_1. \tag{2.22}$$

An example of a linear interpolated TIN is shown in figure 2.22.

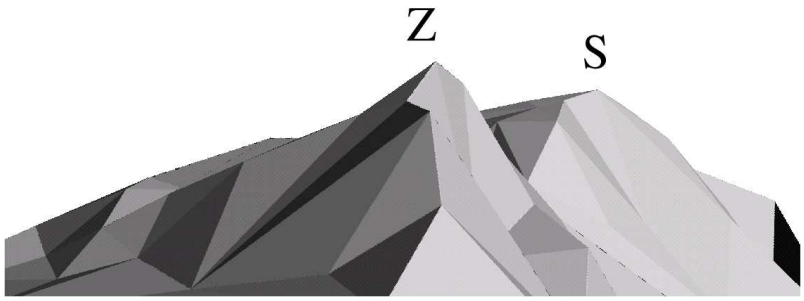

Figure 2.22: Linearly interpolated TIN

Discussion

Linear interpolation is mathematically simple. Therefore, many methods to directly derive advanced information from linearly interpolated TINs (also called tetrahedral terrain) have been developed. Examples are the computation of the visible area (Floriani and Magillo, 1996) or the extraction of hydrological features (Martinoni, 1997). The property that local extrema do not occur within a triangle makes linear interpolation the preferred method for hydrological modeling with TINs. Spurious sinks may still happen because of triangle edges crossing valleys, but these can easily be recognised and the

affected triangle edges can be swapped. Swapping a triangle edge implies finding the quadrilateral formed by the two triangle facets separated by this edge and to swap the edge such that it is connected with the other two vertices of the quadrilateral. Edge swapping can only be trivially achieved if the quadrilateral concerned is convex. In the case of concave quadrilaterals, repeated swapping operations involving neighbouring triangles are required to clean up the situation (Weibel and Brändli, 1995).

Linear interpolation over triangles introduces artifacts: planar facets and a breakline at each triangle edge. Even in terrain types which are very rugged, most breaklines in a linear triangulation are at the wrong locations. Because of the planar facets, curvature is zero everywhere. At triangle edges, surface normals are not defined.

2.7.3 Bivariate quintic interpolation

Bivariate quintic interpolation is a method for G_1-continuous interpolation over triangles. For each triangle facet, a quintic polynomial with 21 coefficient is specified (Akima, 1978; Heller, 1990). The value of an interpolation point can thus be calculated using the following formula:

$$z = \sum_{m=0}^{5} \sum_{n=0}^{5-m} q_{mn} l_1^m l_2^n, \tag{2.23}$$

where l_1 and l_2 are two *barycentric coordinates* of the interpolation point and q_{mn} are the coefficients (Preusser, 1984).

To set up an equation system to solve for the 21 coefficients, the same number of conditions can be specified. 3 conditions are the z-values at the three triangle vertices. The two first partial derivatives with respect to x and y at each vertex are 6 more. 9 conditions are given by the 3 second derivatives at the vertices. The remaining 3 conditions can be specified by the constraint that the global surface has continuous surface normals at the triangle edges. Preusser (1984) gives formulae to calculate the coefficients.

Values for surface derivatives are usually not given as part of the input data and thus need to be estimated. The properties of a surface depend considerably on these estimations.

Bivariate quintic interpolation has been implemented in ARC/INFO in the TINLATTICE command (ESRI, 2002). An extension to consider breaklines is also implemented in this software package. 3D plots show that this breakline extension exhibits similar properties and artifacts as the breakline extension described in chapter 4 for the Coons interpolation and in chapter 5 for the Clough-Tocher Bézier splines. Unfortunately, no mathematical details of this extension have been published (ESRI, 2002). Furthermore, visual inspection suggests that the surface may not be G_0-continuous along the breaklines. Perspective views of the test data set generated by bivariate quintic interpolation are provided in Figures 2.23 and 2.24.

Bivariate quintic interpolation has been used in some applications, e. g. prediction of alpine vegetation (Dirnböck et al., 2003), morphometric relief classification (Unbenannt, 1999) or photogrammetry (Song and Haithcoat, 2003). Dirnböck et al. (2003) and Unbenannt (1999) used input data with breaklines, thus the breakline extension was important. Song and Haithcoat (2003) mentioned the property of smoothly varying normals and the related removal of artifacts due to the planar facets in linear interpolation as being crucial. An observation not only valid for bivariate quintic interpolation is that the implementation within one of the major GIS software packages (ARC/ INFO in the case of bivariate quintic interpolation) is a necessary prerequisite for an interpolation method to be used by a large community.

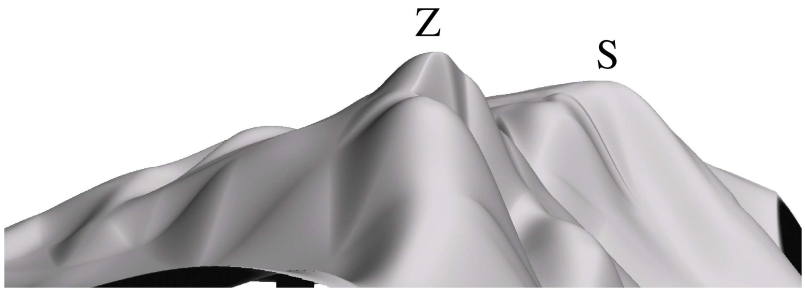

Figure 2.23: Bivariate quintic interpolation without consideration of breaklines.

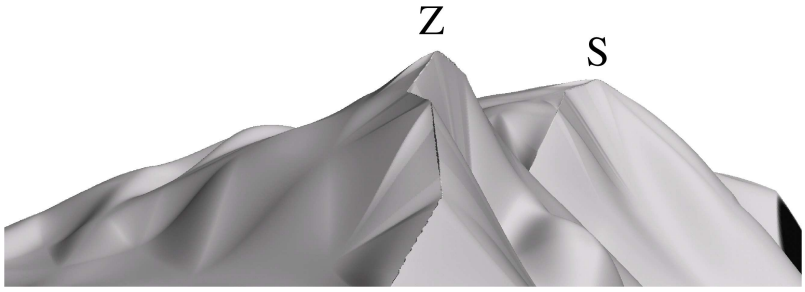

Figure 2.24: Bivariate quintic interpolation with breakline.

Discussion

Bivariate quintic interpolation has many advanced properties:

- It models the curvature of terrain.
- It has continuously varying elevation and surface normals over the whole triangulation.
- Smooth as well as sharp linear features can be modelled.

However, the polynomials are of degree five and may therefore undulate (Schneider, 1998). In the chapters 3 and 5 methods using polynomials of degree three with similar advantages to those mentioned above are introduced. Bivariate quintic interpolation requires the estimation of 5 parameters for each vertex (the partial derivatives up to order two), which is more than with most other methods.

The assessments of bivariate quintic interpolation in the literature are conflicting. Goodchild (1992) sees its only advantage in creating visually pleasing surfaces and not in accurate depiction of terrain. Schneider (1998) on the other hand states that G_1-continuity of a terrain model, with the exception of explicitly specified breaklines, is necessary for geomorphological plausibility.

2.7.4 Clough-Tocher Bézier splines

Bézier surfaces are parametric surfaces described by means of control points (Farin, 1997). Cubic Bézier triangles can be used to interpolate the triangular facets of a TIN. To achieve G_1-continuity, each Bézier triangle must be subdivided into three subtriangles. This method is known as Clough-Tocher interpolant (Farin, 1997). As this method is important in this thesis, detailed introductions to Bézier triangles and the Clough-Tocher split are given in Chapter 5. Figure 2.25 shows a perspective view of a surface interpolated with Clough-Tocher Bézier splines. Schneider (1998) introduced Clough-Tocher splines for the use in digital terrain modeling.

Discussion

Clough-Tocher splines have many interesting properties for surface interpolation. As with bivariate quintic interpolation, global G_1-continuous interpolation is possible. Because the individual triangle surfaces are only cubic, Clough-Tocher splines do not undulate as much as bivariate quintic interpolation (Schneider, 1998).

Clough-Tocher splines require the estimation of normals at the vertices of a TIN. Additionally, for each edge of a TIN, a parameter has to be estimated describing the behaviour of the cross-derivatives along the edge. The estimation of this parameter is thus a crucial task for interpolation. However, the number of parameters to be estimated is considerably less than with bivariate quintic interpolation.

Although Clough-Tocher splines seem to be well suited to represent breaklines, no modification of this scope can be found in the literature.

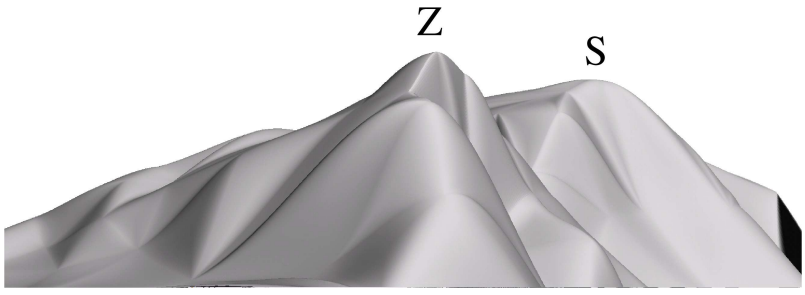

Figure 2.25: Interpolation with Clough-Tocher Bézier splines (linearly interpolated cross-derivatives).

2.8 Summary

In this chapter, interpolation techniques for scattered points used within GIS have been reviewed. They can be classified into two groups: point based methods (inverse distance weighting, kriging, minimum curvature splines) and methods based on a tessellation (finite elements, linear interpolation, bivariate quintic interpolation and Clough-Tocher Bézier splines). Because of the tessellation based approach and local shape control, triangle based methods seem well suited to consideration of linear terrain features. Therefore, in this thesis, triangle based methods are used. However, for cubic interpolation methods, no method for breakline insertion exists. Furthermore, artifacts of the triangulation exist, especially when many linear features are present. The goals of this thesis are therefore to apply Coons patches and Clough-Tocher Bézier splines to terrain modeling, to extend these methods with the possibility of inserting breaklines and to avoid artifacts of the triangular tessellation.

In Table 2.1 properties of the methods reviewed in this chapter are compared. These include the necessity to create a triangulation, the ability to include breaklines and structure lines, the computational cost, the ease of use, and the geometric continuity properties.

Table 2.1: Properties of the reviewed interpolation methods

method	tessellation required	breaklines	structure lines	computational cost	ease of use	continuity
IDW	no	no	no	high O(n^2) range reduces the computational cost significantly	one parameter needs to be estimated	$\geq G_0$
Kriging	no	no	no	high O(n^3) for points within the range	variogram estimation required	$\geq G_0$
MC spline	no	no	no	high O(n^3), but subdivision possible	full automation possible	$\geq G_2$
Finite Elements	no	yes, with bilinear inter-polation	no	high, although sophisticated algorithms can be used for matrix inversion	size of the elements needs to be chosen	G_0/ G_1
Linear Triangles	yes	yes	no	triangulation: O($n \log n$) fast computation afterwards	full automation possible	G_0
Bivariate quintic	yes	yes	yes	triangulation: O($n \log n$) relatively fast computation afterwards	full automation possible	G_1
Clough-Tocher	yes	yes	yes	triangulation: O($n \log n$) relatively fast computation afterwards	full automation possible	G_1

Chapter 3

Interpolation of continuous surfaces for terrain modeling with Coons Patches

3.1 Coons patches and terrain modeling

The Coons patch method was originally developed by S. Coons in the mid 1960s for use in the car industry (Piegl and Tiller, 1997). While the original Coons patch has a rectangular shape, the method has since been generalised for the use in triangular patches (Farin, 1997).

We suggest that Coons patches may be useful for many applications using terrain models. They are very flexible, the shape can be influenced by the curve network, by the blending functions (see section 3.2) as well as by specification of derivative functions (theoretically up to any desired degree) orthogonal to the curves. It is straightforward to imagine the effect of these parameters on the shape of the surface, which is an advantage for the inclusion of additional knowledge in the interpolation. The inclusion of breaklines into a surface is facilitated because, for example, the cross derivative function of a boundary curve is a parameter.

3.2 Method

3.2.1 Basic principle of Coons Patches

The basic principle of Coons patches is to interpolate a surface which fits a set of boundary curves and their cross derivatives. These cross derivatives can then be used, together with the derivative of the boundary curves, to derive normal vectors to the boundary curves. Since neighbouring patches share the same boundary curve and cross derivative, the surface itself is G_1 continuous as any point on the surface, including a point which lies on a boundary curve, has a unique value for its normal.

The steps in interpolating a surface using Coons patches are as follows:

- Build a triangular tessellation.
- Estimate normals at the data points.
- Specify a network of curves bounding the tessellation using cubic Bézier curves.
- Then, for every triangle defined by a set of boundary curves we can interpolate the value of any point within the triangle through the following steps:

– Map the vertices and the boundary curves of the triangle to the standard triangle (a right-angled triangle).
– For the elevation values we calculate three 'ruled surfaces', from which we can derive an elevation value for the Coons patch itself.

Figure 3.1 shows a workflow for this process. In the examples shown here (where we visualise a surface) a dense raster of elevation values is interpolated in every triangle. Each triangle contains of the order of hundreds of such points. If we simply wished to know an individual elevation value (or its normal) within a triangle, this operation would only be carried out once.

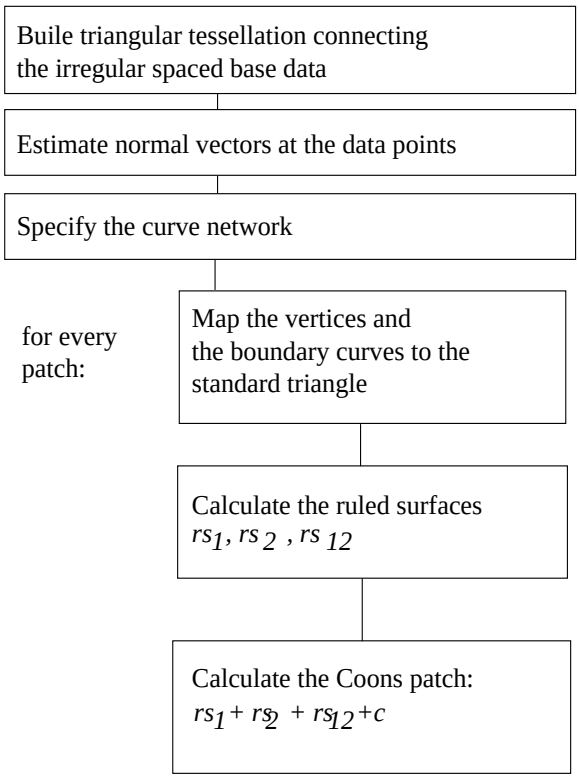

Figure 3.1: Workflow for the specification of a surface with triangular Coons patches.

3.2.2 Estimating normals at data points

The simplest way to estimate surface normals at the data points in a triangular tessellation is to average the surface normals to the *x*,*y*,*z*-planes formed by linearly interpolating between the tessellation boundaries. Woo et al. (1999) demonstrate how these normals can be weighted using the angles of each triangle at the data point (whereby triangles with large angles are considered to have greater weights in deriving the normal).

3.2.3 Specifying the curve network

In this chapter cubic Bézier splines are used (see Chapter 5) to model the segments of the curve network. Each edge of a triangle is a straight line with respect to the *x* and *y*-coordinates, but a

curve with respect to x, y, z-coordinates. Since the surface being derived is $G1$-continuous every point on these segments must have exactly one surface normal. At the data points the surface normals are estimated as described above (3.2.2). The corresponding curves must fit these normal values. Since we are using Bézier splines a further two control points on the curve are required. These are positioned to divide the the segment in the *x*,*y*-plane into three equal lengths. The elevation at these points (corresponding to their position in the *z* plane) is calculated as the intersection of the segment orthogonal to the normal of the adjacent data point and the segment perpendicular to the triangle edge with respect to the *z* axis (Figure 3.2).

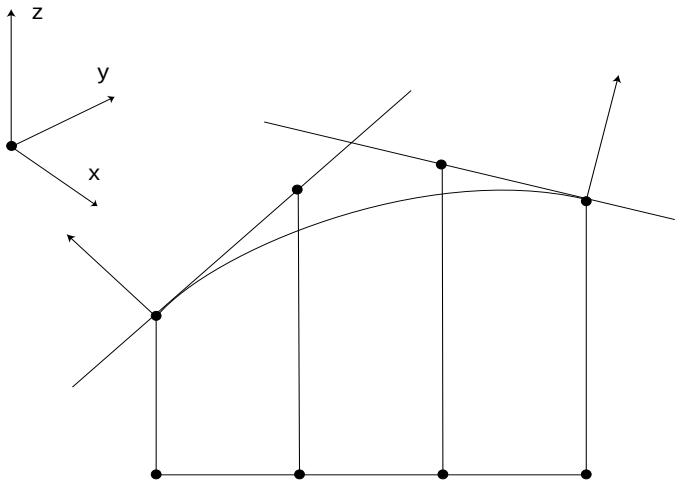

Figure 3.2: Construction of a segment of the curve network.

3.2.4 The standard triangle

According to Barnhill and Gregory (1975) and Klucewicz (1978), the standard triangle, that is a right angled triangle with the right angle being at the origin of a local coordinate system, is used in the calculation of the Coons patch. This triangle is expressed in terms of coordinate axes p and q and we must transform from the *x*,*y*-coordinate plane to the *p*,*q*-coordinate system. Our data and control points are simply mapped from *x*,*y*-space to *p*,*q*-space. To transform any point within the triangle we use the following transformation (modified from Klucewicz (1978)):

$$p = \frac{(x_1 - x_3)(y_3 - y) + (y_1 - y_3)(x - x_3)}{(x_2 - x_3)(y_1 - y_3) - (x_1 - x_3)(y_2 - y_3)}, \tag{3.1}$$

$$q = \frac{(y_2 - y_3)(x_3 - x) + (x_2 - x_3)(y - y_3)}{(x_2 - x_3)(y_1 - y_3) - (x_1 - x_3)(y_2 - y_3)}. \tag{3.2}$$

where p and q are the transformed coordinates of the point (x,y) and (x_1,y_1), (x_2,y_2) and (x_3,y_3) are the coordinate of the data points of the triangle (Figure 3.3).

We also need to transform the cross-derivatives into the *p*,*q*-space. At the data points this is done by defining a point on the derivative to the data point. We then transform this point into *x*,*y*-space using equations 3.1 and 3.2.

To calculate each ruled surface we need the cross-derivative of the segments which are both perpendicular to the triangle edge and intersect the point (p,q) for whom we wish to interpolate elevation.

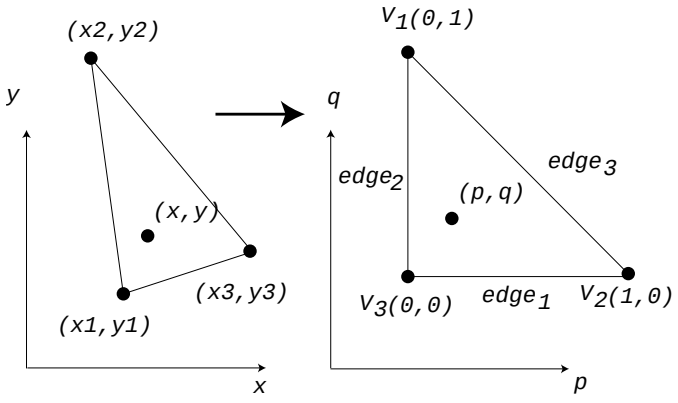

Figure 3.3: Left: Triangle in the *x*,*y*-space. Right: Standard triangle in the *p*,*q*-space.

These cross-derivatives are calculated by interpolating between the transformed cross-derivatives of the data points in the *x*,*y*-plane. Here a linear interpolation is used, but there are other possibilities, e. g. cubic interpolation (Barnhill and Gregory, 1975). Since we wish to define a plane in our transformed *p*,*q*,*z*-space, we calculate a further derivative at the end point of the segment through (*p*,*q*) on the triangle edge in the direction of the boundary curve. The plane defined by these two derivatives then allows us to calculate the cross-derivative of the boundary curve (Figure 3.4). Note that the length of the cross-derivative can be chosen and influences the shape of the surface considerably. If the cross-derivatives are short, the surface is stiffer than it would be with long cross-derivative vectors.

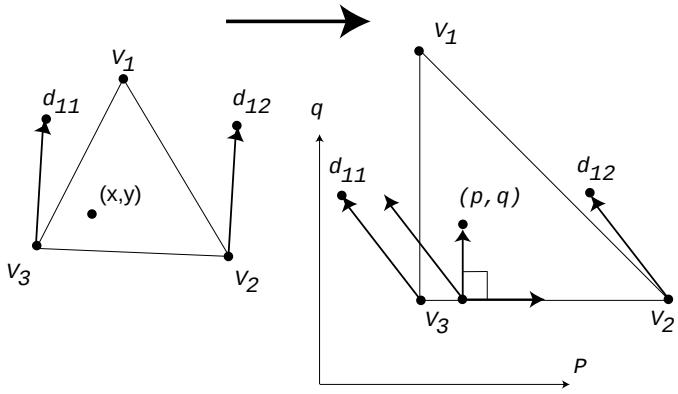

Figure 3.4: Calculation of the cross derivative vector *cd*1 in *p*,*q*-space.

3.2.5 Ruled surfaces

Three ruled surfaces and a correction term can now be derived. The Coons patch itself can then be calculated as an arithmetic expression of these surfaces:

$$coons = rs_1 + rs_2 - rs_{12} + c \tag{3.3}$$

where

rs_1 is the ruled surface defined by edges 2 and 3 and their cross-derivatives;
rs_2 is the ruled surface defined by edges 1 and 3 and their cross-derivatives;
rs_{12} is the ruled surface defined by edge 3 and the cross-derivative of ruled surface 1 at edge 2; and
c is a correction term.

In this paper, cubic Hermite blending functions are used to calculate rs_1, rs_2 and rs_{12}. In the cubic case, a Hermite curve is defined by the two endpoints p_0, p_1 and the two derivative vectors m_0, m_1 at the endpoints (Farin 1997):

$$p(t) = p_0 H_0^3(t) + m_0 H_1^3(t) + m_1 H_2^3(t) + p_1 H_3^3(t). \tag{3.4}$$

The $H_j^n(t)$ are the cubic Hermite polynomials:

$$\begin{aligned} H_0^3(t) &= B_0^3(t) + B_1^3(t), \\ H_1^3(t) &= \frac{1}{3} B_1^3(t), \\ H_2^3(t) &= -\frac{1}{3} B_2^3(t), \\ H_3^3(t) &= B_2^3(t) + B_3^3(t). \end{aligned} \tag{3.5}$$

and the $B_j^n(t)$ are Bernstein polynomials (Section 5.2).

Equation 3.6 shows the adaption of this formula for the calculation of the first ruled surface (rs_1); rs_2 is similar.

$$rs_1 = p_2 H_0^3(\frac{p}{1-q}) + cd_2 H_1^3(\frac{p}{1-q}) + cd_3 H_2^3(\frac{p}{1-q}) + p_3 H_3^3(\frac{p}{1-q}). \tag{3.6}$$

p_2 and p_3 are the points at edge2 and edge3 with coordinate q, cd_2 and cd_3 the corresponding cross-derivatives.

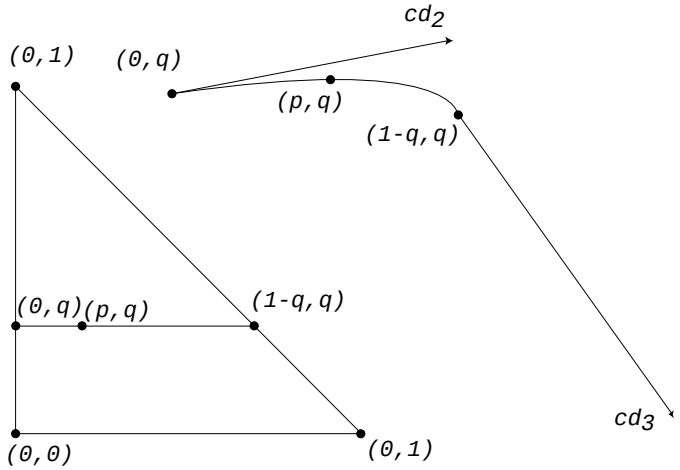

Figure 3.5: Calculation of the point (*p*,*q*) on the ruled surface 1. Left: standard triangle, right: profi le of the line from $(0,q)$ to $(1-q,q)$.

rs_{12} is defined by the following equation:

$$rs_{12} = rs2(0,q) H_0^3(\frac{p}{1-q}) + \frac{\partial rs_2}{\partial p}(0,q) H_1^3(\frac{p}{1-q}) + cd_3 H_2^3(\frac{p}{1-q}) + p_3 H_3^3(\frac{p}{1-q}). \tag{3.7}$$

Finally, a correction term c is needed. It is defined as follows (Barnhill and Gregory, 1975).

$$c = \frac{-p^2q(p+q-1)^2}{p+q}[(\frac{\partial}{\partial q}(\frac{\partial z}{\partial p}))(0,0) - (\frac{\partial}{\partial p}(\frac{\partial z}{\partial q}))(0,0)]. \tag{3.8}$$

3.3 G_1-Continuity

A triangle based, globally G_1-continuous surface must be

- G_1-continuous at the data points;
- G_1-continuous across each triangle edge.

It is possible to use any form of boundary curves with Coons patches. To have a concrete example, cubic Bézier splines are used in this section. A surface is G_1-continuous at a data point P_i if and only if the Bézier control points adjacent to the data point and the data point itself are coplanar. This is achieved with the estimation and usage of the normal vector $\vec{n}_i$ as described in 3.2.2. Additionally, surface derivatives at P_i orthogonal to the triangle edges have to be determined which are later used to specify the shape of the Coons patches along the triangle edges. These cross-boundary derivatives need to conform to the surface normal $\vec{n}_i$, that is, they must result in tangential planes to the surface patches at P_i that are identical to the plane defined by $\vec{n}_i$. As explained in section 3.2.4, the endpoints of the cross-derivative vectors, d_{11} and d_{12} for the first boundary curve, d_{21} and d_{22} for the second one and d_{31} and d_{32} for the third one, are used.

Figure 3.7 shows a triangle with its Bézier spline boundary curves. P_i, $i = 1 \ldots 3$ are the corner points of the triangle. P_3, C_{32}, C_{21} and P_2 are the Bézier control points of the first boundary curve b_1, P_1, C_{12}, C_{31} and P_3 are the control points of b_2, and P_1, C_{11}, C_{22} and P_2 are the control points of b_3. In order to ensure G_1-continuity with adjacent triangles, the Bézier control points C_{31} and C_{32} have to lie in the plane orthogonal to $\vec{n}_3$ passing through P_3. Analogously, C_{12} has to be in the plane defined by $\vec{n}_1$ and P_1, C_{21} in the plane defined by $\vec{n}_2$ and P_2. As a result, all surface patches having a triangle corner point P_n in common, have the same tangential plane at this point P_n.

To ensure G_1-continuity across triangle edges the cross-boundary derivatives (which are determined by the estimated normals at the data points) have to be interpolated along an edge in the same way in both adjacent triangles. A common way to do this is to use linear interpolation of the cross-boundary derivatives (Farin, 1997; Klucewicz, 1978). The cross-boundary derivatives are then used in the Hermite blending of the ruled surfaces. Figure 3.6 shows the test data set Zuestoll interpolated with G_1-continuous cubic Coons patches with linearly interpolated cross-derivatives.

The approach described in this chapter allows for the specification of G_1-continuous surfaces. However, linear features explicitely given as breaklines cannot be considered. As a consequence, sharp features are artificially 'smoothed' not only blurring the topographic features but also distorting the geometry of the immediate neighbourhood. Chapter 4 therefore introduces a method for considering breaklines.

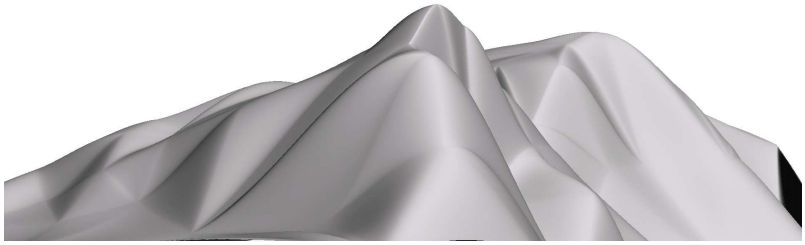

Figure 3.6: Test data set Zuestoll interpolated with G_1-continuous cubic Coons patches using linearly interpolated cross-derivatives.

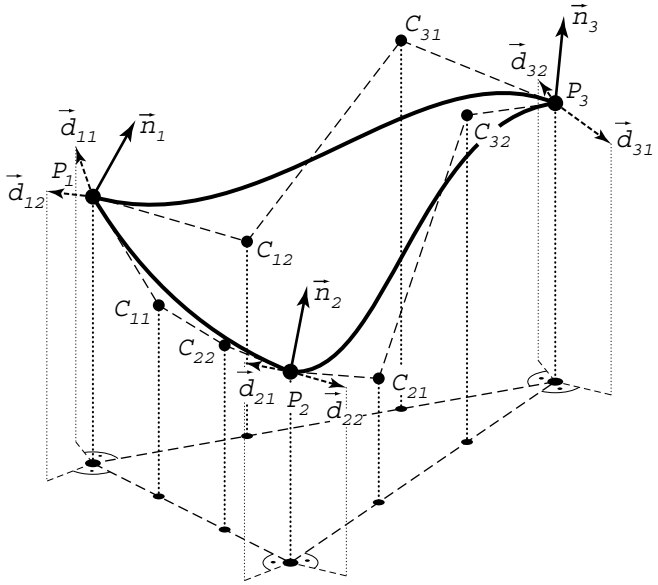

Figure 3.7: Triangular Coons patch with Bézier spline boundary curves.

Chapter 4

Breaklines in Coons surfaces over triangles for terrain modeling

4.1 Inserting breaklines

4.1.1 The problem

Breaklines are lines across which the slope of the surface changes abruptly, that is, across which the surface is not G_1-continuous. They are, for instance, acquired photogrammetrically or digitised from topographic maps. Usually, breaklines are stored as polylines, that is, as ordered sequences of vertices connected by straight lines. For inclusion in surface modeling, such polylines are first inserted into the triangulation of the given elevation data as so-called hard breaklines. In the resulting triangulation, no triangle edge intersects the breakline segments, and each breakline segment is represented by one triangle edge. Commonly, such triangulations are realized as constrained Delaunay triangulations (de Floriani and Puppo, 1992).

Once the breakline segments are represented in the triangulation, G_1-continuity can straightforwardly be abandoned along the breakline edges by specifying different cross-boundary derivatives for the two sides of the breakline segment. For instance, two different derivatives can be estimated *in the middle of each breakline segment*. The derivatives at the endpoints and at the mid-point of the segment are then quadratically interpolated. However, while introducing breaks along the breakline boundary curves, this approach preserves G_1-continuity at the breakline vertices which obscures the topographic structures at these locations (Figure 4.1).

The following sections present an improved method for introducing breaklines to G_1-continuous surfaces specified by means of Coons patches.

4.1.2 Abandoning G_1-continuity

As explained in section 3.2.2, surface normals are estimated at the data points to achieve global G_1-continuity. We will refer to these normals as *ordinary normals* $\vec{n}$. At the breakline vertices, however, the surface is not G_1-continuous, and, thus, tangential planes are not defined. As a result, breakline vertices must be treated differently than ordinary data points.

Breakline segments separate two (or more) regions that are G_1-continuous. Therefore, a partial surface normal is estimated for each region. (Only regular breakline vertices are considered here. Endpoints of breaklines and breakline bifurcations are discussed below.) These vectors $\vec{p}_{i,left}$ and

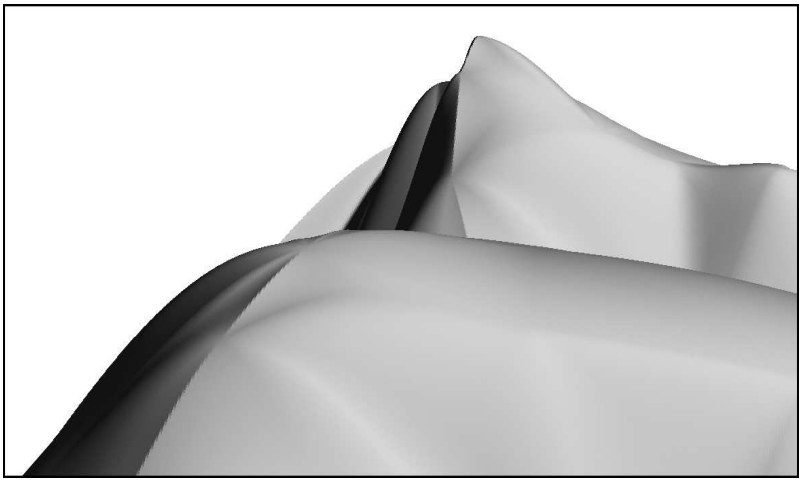

Figure 4.1: Breaklines modelled along the breakline segments but not at the breakline vertices.

$\vec{p}_{i,right}$ (p stands for *partial surface normal*) can be estimated, for instance, by averaging the surface normals of all planar triangle faces adjacent to P_i on the corresponding sides of the breakline.

4.1.3 Ensuring G_0-continuity along patch boundaries

In the general case, G_0-continuity is not realized along the breakline if the boundary curves are specified with the partial surface normals. Figure 4.2 illustrates this problem. The edge between P_1 and P_2 is a segment of a breakline. The boundary curve between P_1 and P_2 is represented by a Bézier spline, which is in turn specified by the control points C_{12R} and C_{22R} for the right side, and by C_{12L} and C_{22L} for the left side of the breakline. These control points are defined by the partial vectors for the right and left side, i.e., by the planes through P_1 and P_2 orthogonal to $\vec{p}_{1,right}$, $\vec{p}_{1,left}$, $\vec{p}_{2,right}$ and $\vec{p}_{2,left}$. Because the partial vectors are different for the right and left side, the normal planes are different. Therefore, C_{12R} is not equal to C_{12L}, and C_{22R} is not equal to C_{22L}. As a result, the boundary curves for the right and left side do not coincide, causing a discontinuity along the breakline.

To avoid this problem, the elevation of the control points along the breakline edges are calculated using the ordinary normals $\vec{n}$ while the elevation of the control points of the remaining boundary curves are calculated using the partial normals $\vec{p}_{i,right}$ and $\vec{p}_{i,left}$, respectively (figure 4.3). In this way, G_0-continuity along the breakline is guaranteed because the ordinary normals are the same for both sides of the breakline. The surface still forms a break because the partial normals are used for the triangle edges on either side of the breakline.

4.1.4 Cross-boundary derivatives and introduction of artifacts

It has been mentioned above that the cross-boundary derivatives at each data point P_i need to conform to the normal vector $\vec{n}_i$. Figure 4.4 illustrates this requirement and highlights a problem at breakline vertices. The left triangle of Figure 4.4 shows a generic triangle. In order to ensure G_0- and G_1-continuity, Bézier control points adjacent to a data point P_i as well as the cross-boundary derivatives at this point must be coplanar. Therefore, these elements are determined by means of the following steps (taking data point P_2 in figure 4.4 as an example):

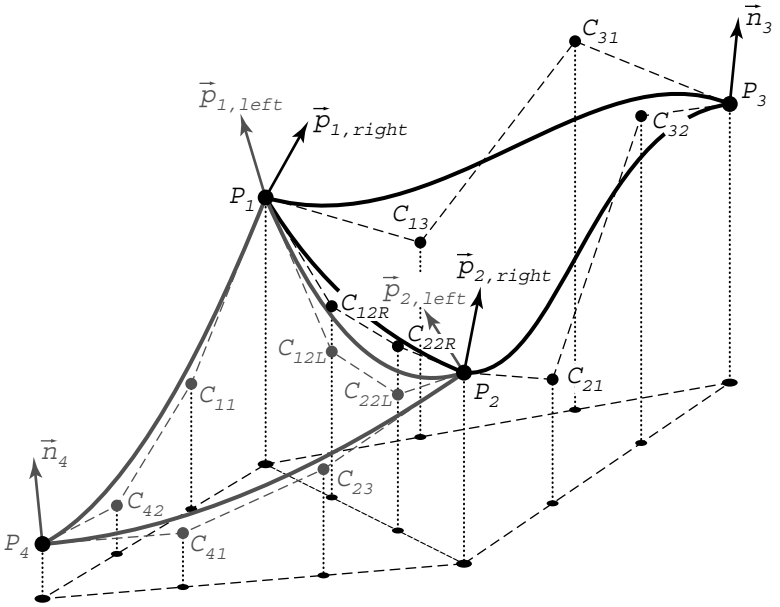

Figure 4.2: Patches left and right of a breakline segment with incompatible partial vector pairs $\vec{p}_{1,right}$, $\vec{p}_{1,left}$, and $\vec{p}_{2,right}$, $\vec{p}_{2,left}$.

1. Normal vector $\vec{n}_2$ is estimated at P_2.

2. The Bézier control points C_{21} and C_{22} are determined so that they lie in the plane through P_2 orthogonal to $\vec{n}_2$, that is, in the tangential plane $\mathcal{N}_2$.

3. The cross-boundary derivatives for all edges around P_2 are determined so that they lie in $\mathcal{N}_2$. In Figure 4.4, only the edges b_{12} and b_{23} are shown. The according cross-boundary derivatives are drawn as vectors $\vec{d}_{21}$ – specifying slope and aspect of the surface at P_2 orthogonal to b_{23} – and $\vec{d}_{22}$ – specifying slope and aspect at P_2 orthogonal to b_{12}.

This simple and effective sequence is not applicable at breakline vertices. In Figure 4.4 right, boundary curve b_{12} is a breakline segment. Consequently, a partial normal vector $\vec{p}_{2,left}$ is estimated, and Bézier control point C_{21} is determined so that is lies in the corresponding plane $\mathcal{P}_{2,left}$. However, the Bézier control point C_{22} of the breakline segment needs to be determined with the help of $\vec{n}_2$ for the reasons discussed in the previous section. As a result, C_{22} does not lie in $\mathcal{P}_{2,left}$ – and C_{12} does not lie in $\mathcal{N}_2$. If the cross-boundary derivatives were determined using either of the two tangential planes, one of the Bézier control points would violate the required coplanarity. This would result in losing not only G_1- but even G_0-continuity. Thus, the cross-boundary derivatives cannot be determined with the help of one of the tangential planes.

Obviously, coplanarity of the Bézier control points and the cross-boundary derivatives is crucial. Therefore, we let the Bézier control points and the data point define the corresponding plane. In Figure 4.4, the cross-boundary derivatives are determined so that they lie in the plane defined by P_2 and the Bézier control points and C_{12} and and C_{22}. This approach ensures G_0-continuity. However, it also means that the cross-boundary derivatives at P_2 are different for the left and right side of each boundary curve. Specifying different cross-boundary derivatives $\vec{d}_{22}$ for the triangle patch on the left of b_{12} than for the triangle on the right side of b_{12} results in the surface not being G_1-continuous along

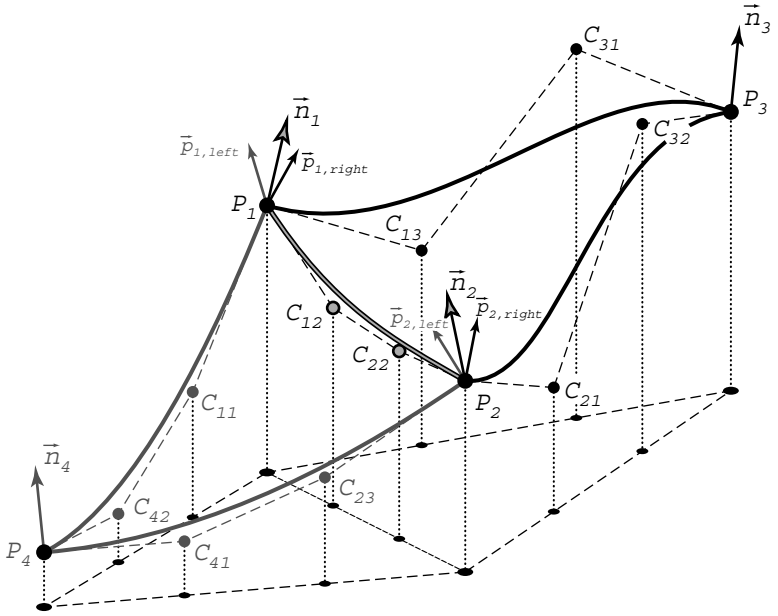

Figure 4.3: Same situation as in Figure 4.2. The control points of the breakline segment, however, are specifi ed by means of the ordinary normal vectors $\vec{n}_1$ and $\vec{n}_2$.

this boundary curve. In the case of the breakline segment b_{12}, this loss of G_1-continuity is deliberate (as b_{12} is supposed to be a breakline).

Likewise, the cross-boundary derivative $\vec{d}_{21}$ at P_2 will be different for the triangle patch to the left of boundary curve b_{32} from the derivative to the right of b_{32}. Again, G_1-continuity is lost along b_{32}. This time, however, this loss is unintentional, and the resulting break in the surface along b_{32} is an artifact (Figure 4.8). This artifact is considered a compromise resulting from the need to use $\vec{n}_2$ to specify the Bézier control point C_{22}.

The surface forms breaks along the three edges of all triangles that are bound by at least one breakline segment. Hence, in figure 4.5 where the boundary curves b_{12}, b_{23}, and b_{34} represent breakline segments, the boundary curves b_{15}, b_{25}, b_{35}, b_{36}, and b_{46} form unintentional breaks. It is worth noting that not all boundary curves leading to a breakline vertex are breaks in the surface. If neither of the two adjacent triangles of a boundary curve b_{ik} at a breakline vertex is bound by a breakline segment, the surface is G_1-continuous along b_{ik}.

4.1.5 Breakline endpoints

Endpoints of breaklines can be handled in different ways. If the surface at the breakline endpoint has to be G_1-continuous, then it is treated as a regular data point. Only the normal vector $\vec{n}_i$ is used for specification of control points and cross-boundary derivatives, partial vectors are not calculated. In this case, the surface break disappears between the second to last breakline point and the endpoint. If the surface at the breakline endpoint has to be G_0-continuous, the boundary curves leading to it have to be specified such that their derivatives are distinct at the breakline endpoint.

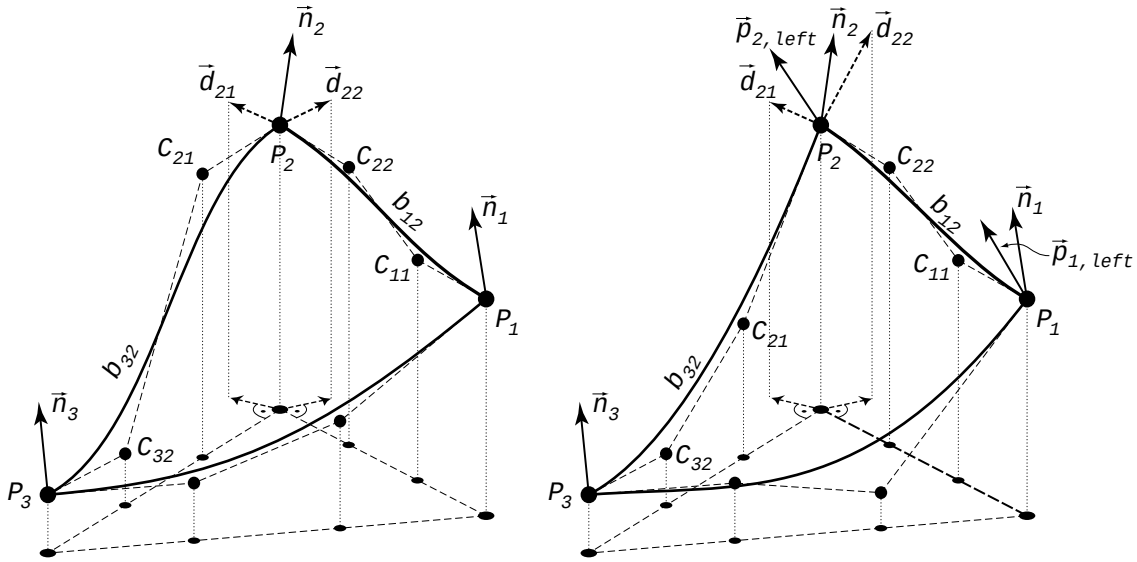

Figure 4.4: Problems with G_0-continuity. Left: Regular triangle where first Bézier control points and cross-boundary derivatives are in the plane orthogonal to $\vec{n}_2$. Right: The planes orthogonal to $\vec{n}_2$ and $\vec{p}_{2,left}$ cannot be used for determination of cross-boundary derivatives.

4.1.6 Breakline bifurcations

For breakline bifurcations, more than two partial normal vectors need to be calculated. Around such a vertex, each sector bounded by two breakline segments is assigned its own partial normal. These partial normals are used instead of $\vec{p}_{1,right}$ and $\vec{p}_{1,left}$. Otherwise, the described method to calculate the elevation of the control points and the new partial normals remains the same.

4.1.7 Example

The test dataset Zuestoll consists of 98 scattered points, some of which are located on structure and breaklines. Both, structure lines and breaklines, have been inserted into the triangulation as constrained triangle edges. At the breaklines, G_1-continuity is abandoned with the described method while at the structure lines, G_1-continuity has been preserved. The main ridge from west to east has been inserted as a breakline.

Figure 4.7 shows views from the west to the main ridge without and with modeling the breakline, respectively. Insertion of the breakline renders the ridge sharper and the surface steeper on both sides. The bulk at the location of the breakline is removed. Figure 4.8 shows a view to the north side of the main ridge. The unintentional breaklines at the north side of the breakline are highlighted. The comparison of the two illustrations in Figure 4.8, however, illustrates that the effect of these undesired breaklines on the surface is small.

4.2 Discussion

Specifying Coons patches that compose a continuously differentiable surface requires the prior specification of surface normals at each data point. With the help of these normals, the specification of

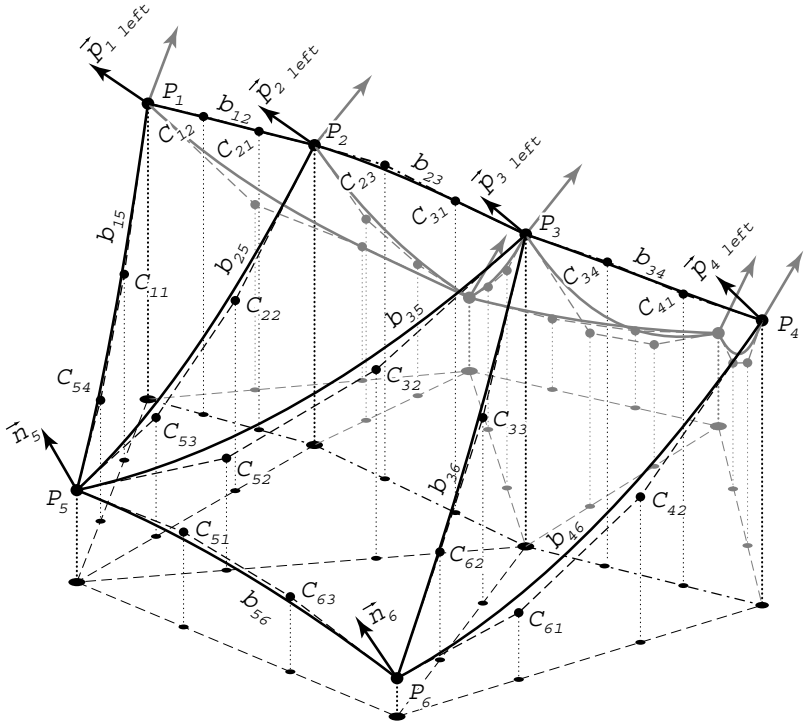

Figure 4.5: Breakline segments and adjacent triangles.

the boundary curves and cross-boundary derivatives is straightforward. Coplanarity of these elements ensures G1-continuity at the data points and along the triangle edges. If a data point is part of a breakline, however, two surface normals need to be estimated, one for each side of the breakline. The control points of the boundary curves have to specified using both normal vectors which makes it necessary to abandon coplanarity of the named elements. The presented method tackles the problem by allowing the formation of additional breaklines adjacent to the breakline vertices. These unintentional breaklines do not disturb the geometry of the surface considerably, and their visual effect is minor.

In the presented approach, the ground plans of breakline segments are straight lines. This forms a severe limitation for two main reasons. First, the true form of the breakline cannot be represented in the digital terrain model. Second, the breakline is not continuously differentiable at its vertices which eventually leads to the introduction of the unintentional breaklines accompanying the actual breaklines. The usage of true 3D Bézier splines (with curved groundplans) seems to be an obvious solution to the problem. However, this approach introduces new obstacles because, for instance, it distorts the parameter space in the triangles adjacent to the breaklines.

This chapter presented a method to model breaklines in a surface specified with the help of Coons patches. Besides Coons patches, the triangular Clough-Tocher Bézier spline is another method which allows for the specification of G_1-continuous surfaces. Although this provides less flexibility in the choice of the boundary curves, it is easier to implement than the Coons method. Chapter 5 explains the basic method and develops a method of inserting breaklines into Clough-Tocher Bézier spline surfaces, which is similar to the one presented in this chapter.

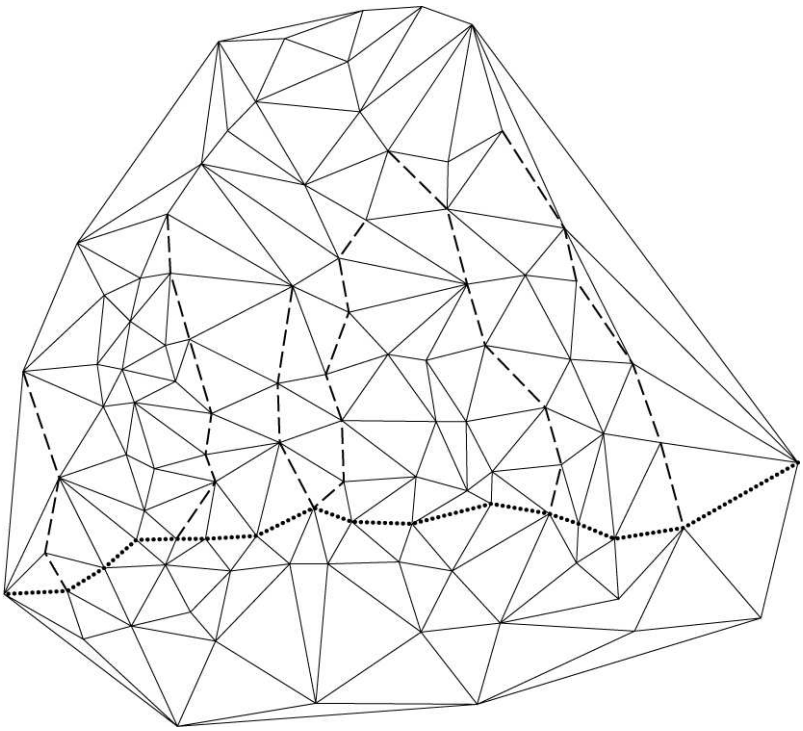

Figure 4.6: Triangulation used for the surfaces visualised in figures 4.7 and 4.8. Breaklines are indicated by a dotted line, structure lines by dashed lines and normal triangle edges by solid line.

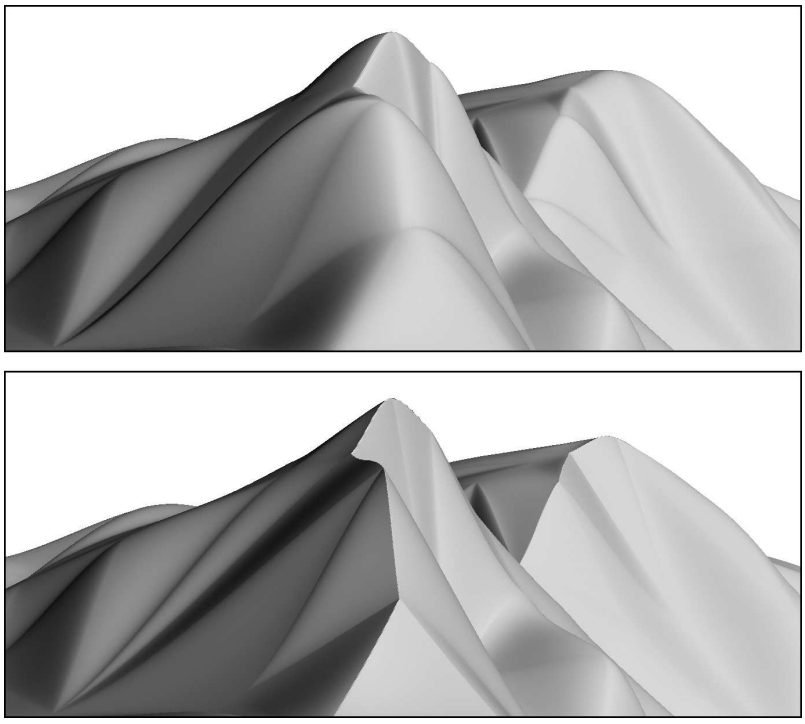

Figure 4.7: Perspective view to the test dataset from the west. The triangulation has been constrained to the ridge. Top: Breakline is not modelled. Bottom: Breakline is modelled.

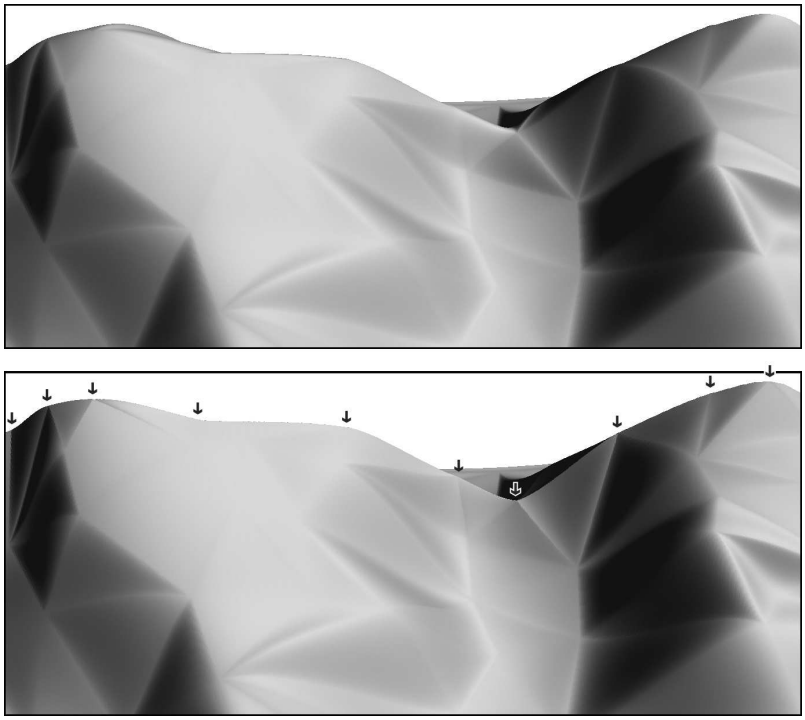

Figure 4.8: Perspective view to the test dataset from the north. The triangulation has been constrained to the ridge. Top: Breakline is not modelled. Bottom: Breakline is modelled; all edges starting at the data points indicated with the arrows form (deliberate or unintentional) breaklines.

Chapter 5

Triangular Clough-Tocher Bézier splines

5.1 Introduction

Bézier curves and surfaces are specified by means of control points. The geometric meaning of the control points is that the curves and surfaces are attracted by them, but in general, they only go through the endpoints (respectively through the vertices for surfaces). The polygon, which is obtained by connecting the control points with straight line segments is called control polygon. Triangular as well as quadrilateral Bézier surfaces exist.

In this thesis, triangular cubic Clough-Tocher Bézier splines are used, as they provide a possibility to generate G_1-continuous surfaces by using elements of cubic degree. Furthermore, the so-called Bernstein form can be used, which proved to be numerically stable as well as computationally efficient (Farin et al., 2002). Because of the Bernstein form and the use of barycentric coordinates, the programming of Clough-Tocher splines is much easier than that of Coons patches.

Schneider (1998) used triangular Clough-Tocher Bézier splines for the interpolation of terrain surfaces. This phd thesis contains an excellent introduction to Bézier curves and surfaces. This chapter starts by explaining the basics of Bézier curves. Then, the generalisation to triangular surface patches and the Clough-Tocher split are shown. The consideration of breaklines is done in a similar way as for the Coons patch. At the end of the chapter, the possibilities to use the remaining degrees of freedom are discussed.

5.2 The Bernstein form of a Bézier curve

The Bernstein polynomials are defined as

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{(n-i)}, \tag{5.1}$$

where the binomial coefficients are given by

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & 0 \le i \le n \\ 0 & else \end{cases} . \tag{5.2}$$

A point on a Bézier curve of degree n can be calculated using a set of $n+1$ control points and Bernstein polynomials:

$$b(t) = \sum_{j=0}^{n} b_j B_j^n(t), \tag{5.3}$$

where the b_j are the control points and t is the parameter of the curve. If t is 0, $b(t)$ is equal to the first control point. If t is 1, $b(t)$ is equal to the last control point. Figure 5.1 illustrates the application of this formula in the case of a cubic Bézier spline. As the Bernstein polynomials used always sum to 1, the Bernstein form is a weighted sum of the control points. Bézier curves have some important properties:

- the first and the last control points are interpolated;
- the curve is always contained in the convex hull of the control points;
- the first derivative at the first control point is the vector from the first control point to the second control point;
- the first derivative at the last control point is the vector from the second to last control point to the last control point;
- no plane intersects a curve more times than it intersects the curve's control polygon. This means that a Bézier curve follows its control polygon rather closely and does not wiggle more than its control polygon.

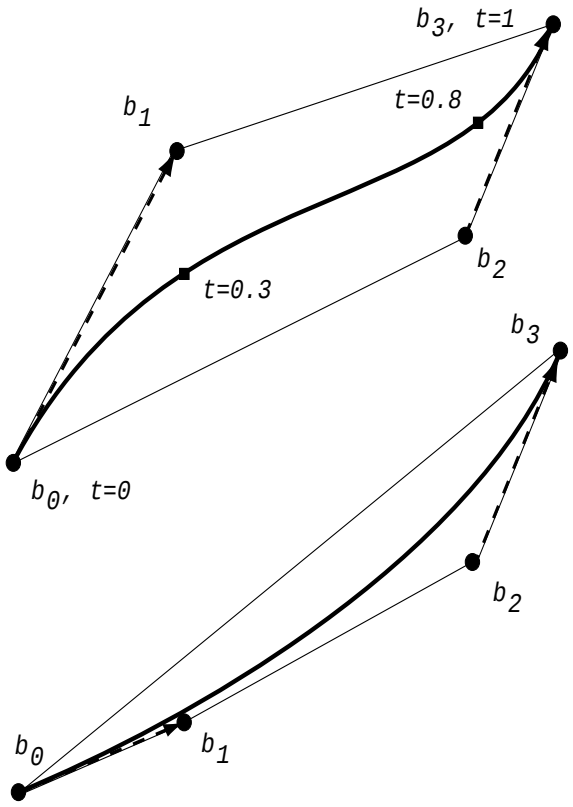

Figure 5.1: Cubic Bézier curves. $b_0 \ldots b_3$ are the control points. The dashed vectors from b_0 to b_1 are the derivatives b_0 and the dashed vectors from b_2 to b_3 are the derivatives at b_3. On the upper curves, the points on the curve for the parameters $t = 0$, $t = 0.3$, $t = 0.8$ and $t = 1$ are marked. For the lower curve, the position of the control point b_1 has been changed. Note that both curves have still the same derivative at b_3.

5.3 Triangular Bézier surfaces

The Bernstein form can also be used to describe surface patches by means of control points. In this section, the principle of triangular Bézier patches is explained and illustrated. The use of barycentric coordinates and bivariate Bernstein polynomials allows for an elegant formulation (Farin, 1997).

5.3.1 Barycentric coordinates

The position of a point within a triangle can be described using barycentric coordinates. Let a, b and c be three vertices defining a triangle. The barycentric coordinates u, v and w of the point p with respect to a, b and c have the property, that

$$p = ua + vb + wc \tag{5.4}$$

and

$$u + v + w = 1. \tag{5.5}$$

The barycentric coordinates of p can be calculated as follows:

$$u = \frac{area(p,b,c)}{area(a,b,c)}, \tag{5.6}$$

$$v = \frac{area(a,p,c)}{area(a,b,c)}, \tag{5.7}$$

$$w = \frac{area(a,b,p)}{area(a,b,c)}. \tag{5.8}$$

Figure 5.2 shows isolines for the barycentric coordinates as well as some points with their barycentric coordinates.

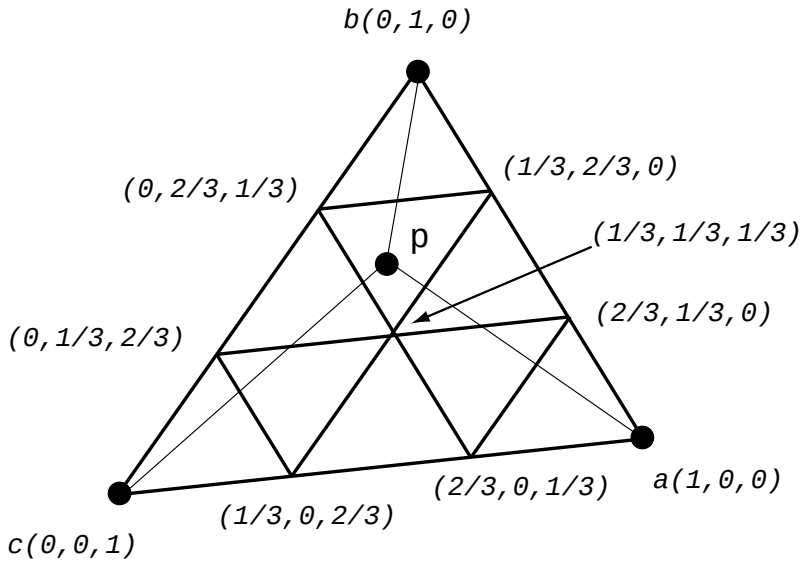

Figure 5.2: The barycentric coordinates of p are the ratios of the subtriangles (indicated by thin lines) to the macrotriangle. The barycentric coordinates of the locations, where the control points for cubic Bézier triangles usually are located, are given.

5.3.2 Bivariate Bernstein polynomials

The bivariate Bernstein polynomials are defined as:

$$B^n_{i,j,k}(u,v,w) = \begin{cases} \frac{n!}{i!\,j!\,k!} u^i v^j w^k & 0 \leq i,j,k \leq n \\ 0 & otherwise \end{cases} \tag{5.9}$$

i, j, k are indexes, equivalent to i in the univariate case (figure 5.3). Note that, even if three coordinates and indexes are used, the polynomials are bivariate, as the barycentric coordinates sum up to 1 and a coordinate is determined if the two others are given.

5.3.3 Cubic Bézier triangles

For the application of the Bernstein form in triangular Bézier patches, as it is used in this thesis, the control points need to be arranged and indexed in a scheme, such that the triangular patch itself is composed of subtriangles. Figure 5.3 shows a cubic triangle and its control points. Each control point has three indexes i, j, k, which always sum up to three in the cubic case. Note that the boundary curves of the triangle do not necessarily have to be straight lines in the orthogonal projection. However, if they are not, the determination of the barycentric coordinates of a point with given x, y and z-coordinates is a difficult task, requiring approximation, such as triangular Bézier clipping (Roth et al., 2000). Therefore, in this thesis, only triangles, which have straight boundary curves in the orthogonal projection, are used.

If the control points are given, a point on a surface can be calculated using the Bernstein form by summing up all the control points, weighted with the Bernstein polynomials:

$$b^n = \sum_{i+j+k=n} b_{i,j,k} B^n_{i,j,k}(u,v,w), \tag{5.10}$$

where u, v and w are the barycentric coordinates and $b_{i,j,k}$ the control points. Similar to the onedimensional case, the Bernsteinpolynomials B involved sum up to 1.

the properties of Bézier triangles are similar those of the curves:

- the plane through a triangle vertex and the two adjacent control points on the boundary curves are directional derivatives on the surface
- the surface is contained in the 3D convex hull of the control points
- the three boundary curves of the triangle are Bézier curves of degree n. To describe one such boundary curve, the $n+1$ control points on the boundary are sufficient.
- Only the control points adjacent to a vertex influence the surface derivatives at this vertex, that is to say, if a surface normal is estimated at a vertex, the elevation values of the two adjacent control points have to be in the plane defined by the estimated normal. When using cubic Bézier triangles, a normal can be estimated at every data point, because applying estimated normals as described does not affect the surface derivatives at the other data points.

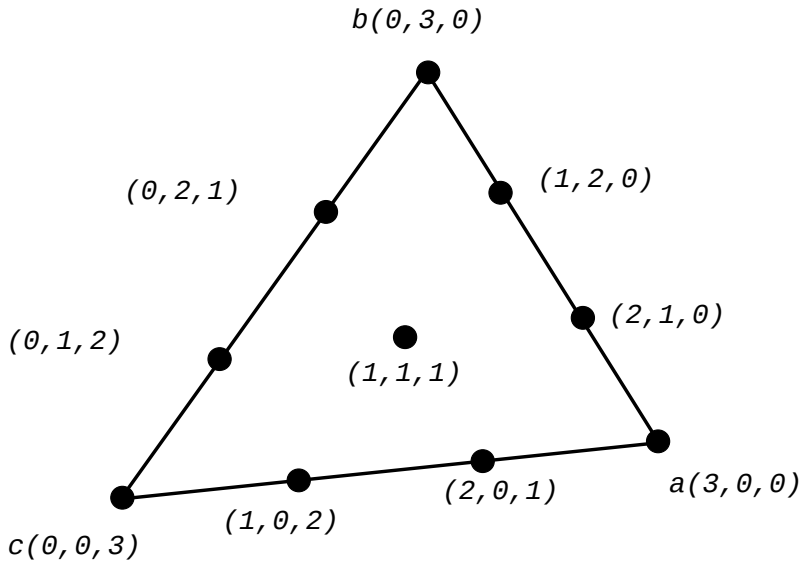

Figure 5.3: Indexes *i, j, k* of the control points of a cubic Bézier triangle.

5.3.4 Continuity between cubic Bézier triangles

For global surfaces composed of several triangular patches, the geometric continuity between the individual patches is important. Figure 5.4 shows two adjacent cubic Bézier triangles. The two patches are C_1 continuous if the pairs of shaded triangles are coplanar and an affine transformation of the two domain triangles (Farin, 1997). This condition is sufficient also for G_1 continuity. In this thesis, the boundary curves of the triangular splines are used such that the control points are evenly spaced in the orthogonal projection. In this case, it is guaranteed that the shaded pairs of triangles are affine maps of the domain triangles. If the control points are placed in other locations, it is very likely, that the affine transformation condition can not be fulfilled. For this case, which is not of interest in this thesis, Farin (1997) gives conditions for G_1-continuity without C_1-continuity.

Although two adjacent cubic Bézier triangles can easily be made G_1-continuous across the common boundary curve, it is in general not possible to have a global G_1-continuous surface consisting of a large number of triangles. The reason for this is that the coplanarity of the middle pair in figure 5.4 depends on the elevations of the control points in the middle of the triangles (P_4, P_9). All three triangles, which share common edges with the triangle under consideration, need a specific elevation value for this middle control point in order to fulfill the coplanarity constraint for the middle pair of triangles. For the whole triangulation, this system of dependencies in general has no solution.

5.4 The Clough-Tocher subdivision

To achieve a global G_1 continuous surface consisting of cubic Bézier triangles, the Clough-Tocher split can be used. The idea of this method is to insert a point at the weighting point of each triangle and to split the triangle into three subtriangles. Triangular Bézier surfaces are then specified for each subtriangle. The Bézier control net of a global G_1 continuous surface can be built by executing the following steps for each original triangle (Figure **??**):

- split the triangle into three subtriangles at the weighting point.

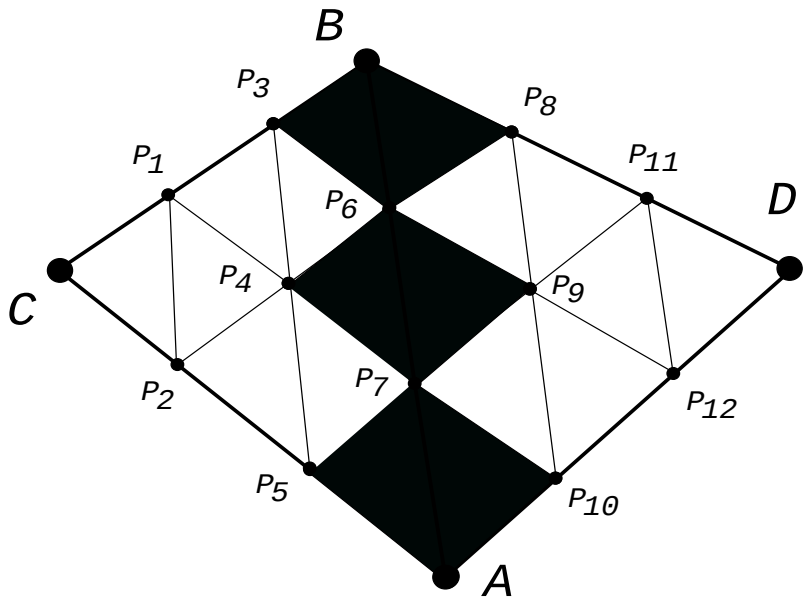

Figure 5.4: Two adjacent cubic Bézier triangles are C_1 and G_1-continuous along the common boundary, if the grey pairs of subtriangles are coplanar and an affine transformation of the domain triangles.

- estimate a tangent plane at the three vertices of the original triangles. This can be done in several ways. The most common method, which is also used in this thesis, is to sum up the normals of the planes defined by the (planar) triangles around a vertex (Watson, 1992). The normals of the surrounding triangles can be weighted. For the prototype software of this thesis, each normal of a planar triangle is weighted by $\frac{1}{d_1^2 d_2^2}$, where d_1 and d_2 are the lengths of the triangle edges, which are connected with the vertex under consideration. Other methods are the use of a least-square function with a subset of neighbouring points or to calculate the gradient from a point based interpolation method, for instance minimum curvature splines (see 2.5) (Watson, 1992).

- set the elevation of the control points P_1, P_6 and P_{12} such, that they are on the estimated tangent plane through A. Do the same for P_2, P_8 and P_3 with respect to the tangent plane at B and for P_4, P_{10} and P_5 with respect to the tangent plane at C.

- Use an assumption to calculate the elevation of the control points P_7, P_9 and P_{11}. An example for such an assumption is linear interpolation of the cross-derivatives at the vertices of the original triangle along a triangle edge. The cross-derivatives at the vertices are given, since tangent planes have been estimated there. In section 5.5, other possibilities will be shown. When creating a new assumption, it is important, that the coplanarity conditions necessary for G_1-continuity are fulfilled. This means that P_1, P_7, P_2, P_{17} have to be coplanar, as well as P_3, P_9, P_4, P_{18} and P_5, P_{11}, P_6, P_{19}. If the elevations of P_7, P_9 and P_{11} are calculated using linear interpolated cross-derivatives, this is guaranteed since the derivatives across an edge are the same on both sides of the edge. Doing the steps described here in the adjacent triangles thus automatically provides G_1-continuity.

- Calculate the elevation of P_{15} such that it is coplanar with P_{12}, P_7 and P_{11}. Similar, P_{13} has to

be coplanar with P_7, P_8 and P_9 as well as P_{14} has to be coplanar with P_9, P_{10} and P_{11}.

- The elevation of the control point in the middle, P_{16} can now be calculated, as it has to be coplanar with P_{13}, P_{14} and P_{15}.

The resulting surface then is G_1-continuous everywhere and G_2-continuous inside a macrotriangle. Figure 5.6 shows a perspective view of the test dataset 'Zuestoll' interpolated with cubic Clough-Tocher Bézier splines using linearly interpolated cross-derivatives.

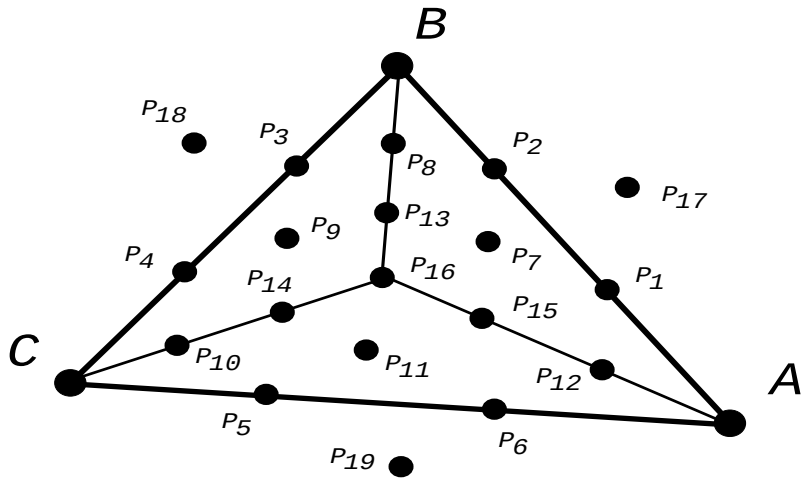

Figure 5.5: Control points of a cubic Clough-Tocher Bézier triangle. *A*, *B* and *C* are the vertices of the original triangle. This triangle is split into three cubic Bézier triangles at the weighting point, P_{16}. P_{17}, P_{18} and P_{19} are the middle control points of the adjacent subtriangles.

Figure 5.6: Test dataset 'Zuestoll' interpolated with Clough-Tocher Bézier splines using linear interpolated cross-derivatives.

5.5 Smoothing Clough-Tocher Bézier splines

In the previous section, the assumption of linearly interpolated cross-derivatives along the triangle edges has been used. Although this assumption is simple, there is no conclusive reason for it. Furthermore, perspective views of surfaces specified with linear interpolated cross-derivatives show interpolation artifacts at the triangle edges (Figures 5.6 and 5.7). The reason for this is that the derivatives across the triangle edges are constrained to the linearly interpolated values. As a result, although the slope of the surface varies continuously, the curvature changes are abrupt across the triangle edges. Therefore, other methods have been used to calculate the elevation of the control points in the middle of the subtriangles.

Schneider (1998) specifies a rectangular cubic Bézier patch over two adjacent macrotriangles. The elevation of the control points of this rectangular patch are calculated using the normal vectors estimated at the data points. The slope between the two control points (of the rectangular patch, cp_1 and cp_2 in Figure 5.8), which are beside the boundary between the triangles is then calculated. The middle control points of the two subtriangles (P_4 and P_9 in Figure 5.8) are then calculated such, that the slope between them is equal to the slope between the control points of the rectangular patch. Together with the constraint that P_4 and P_9 are coplanar with the two control points on the boundary curve between the two macrotriangle, the elevations of P_4 and P_9 are uniquely defined.

Figure 5.7: Upper picture: Clough-Tocher interpolation using linear interpolated cross-derivatives. The artifacts of the cross-derivatives are marked with arrows. These artifacts, which are visible as lines of similar color at the triangle edges, occure because the derivatives across the triangle edges are constrained to the linear interpolated values. Lower picture: smoothed Clough-Tocher spline.

Farin (1985) introduced a method which tries to use the available degrees of freedom, such that the transitions between the triangles come as close to C_2-continuity as possible. First, linearly interpolated cross-derivatives are used to calculate the elevation of all the control points. Then, the middle control points of the subtriangles are calculated such that the difference of the control points to the C_2 situation

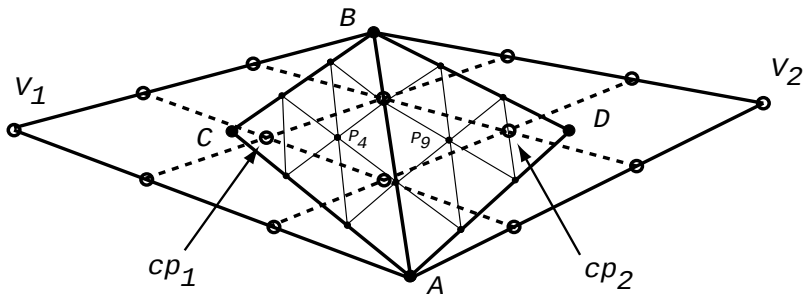

Figure 5.8: Method of [Schneider1998] to determine the elevations of the control points P_4 and P_9. A rectangular cubic Bézier patch is specified with V_1, A, V_2, B as vertices. This patch has 16 control points, including cp_1 and cp_2. The elevations of these two points are given, since cp_1 has to be on the normal plane through V_1. Similarly, cp_2 has to be on the normal plane through V_2. Finally, the elevations of the points P_4 and P_9 are calculated such that the line between these two points has the same gradient as the line from cp_1 to cp_2.

is minimised. Kashyap (1996) extended this method by using an iterative approach. However, this requires the storage of all control points and thus needs much memory, especially in large terrain datasets. Therefore, in this thesis, the method of Farin (1985), with just one iteration, is used. The necessary formulas are derived the sections 5.5.1 and 5.5.2 because some typesetting errors seem to be present in Farin (1985).

5.5.1 C_2 conditions

For a C_2 transition between two Bézier patches, the following two equations have to be fulfilled (Kashyap, 1996):

$$uP_1 + vP_4 + wP_3 - u'P_{11} - v'P_8 - w'P_9 = 0 \tag{5.11}$$

and

$$uP_2 + vP_5 + wP_4 - u'P_{12} - v'P_9 - w'P_{10} = 0, \tag{5.12}$$

where the control points P are those of Figure 5.4. u, v, w are the barycentric coordinates of D with respect to the triangle A, B, C and u', v', w' are the barycentric coordinates of A with respect to the triangle D, B, A.

5.5.2 Lagrange minimisation

In general, these conditions cannot be fulfilled, but the deviation from the C_2 situation can be minimised. The elevations of all the control points can be calculated using linear interpolated cross-derivatives. Then, the elevations of the middle control points are chosen such as to minimise the summed deviations from the C_2 situation. Because the transition should still be C_1-continuous, we need to have the coplanarity of P_4, P_6, P_7 and P_9 as a constraint. Note that we only have to carry out this minimisation for the z-component. So, the points in the following formulas can effectively be replaced by their z-coordinates. In terms of equations, we need to minimise

$$\begin{aligned} F(P_4, P_9) \quad = \quad & (uP_1 + vP_4 + wP_3 - u'P_{11} - v'P_8 - w'P_9)^2 \\ & + (uP_2 + vP_5 + wP_4 - u'P_{12} - v'P_9 - w'P_{10})^2 \end{aligned} \tag{5.13}$$

considering the coplanarity constraint

$$uP_4 + vP_7 + wP_6 - P_9 = 0. \tag{5.14}$$

Using the Lagrange multiplier λ yields the function to minimise,

$$\begin{aligned} F(P_4;P_9;\lambda) &= (uP_1 + vP_4 + wP_3 - u'P_{11} - v'P_8 - w'P_9)^2 \\ &\quad +(uP_2 + vP_5 + wP_4 - u'P_{12} - v'P_9 - w'P_{10})^2 \\ &\quad +\lambda(uP_4 + vP_7 + wP_6 - P_9). \end{aligned} \tag{5.15}$$

Setting the three partial derivatives to zero and solving the resulting equation system with respect to P_9 provides the result:

$$\begin{aligned} P_9 &= (v'u^3P_2 + uwu'P_{12} + w^2vP_7 - wu^2P_2 - vu^2P_1 \\ &\quad -uvwP_3 + w'u^3P_1 + v^2wP_6 - uv'wvP_7 - uwvP_5 \\ &\quad +uvu'P_11 + uvv'zP_8 - u^2v'u'P_{12} + v^3P_7 - uw'v^2P_7 \\ &\quad -uv'w^2P_6 + uww'P_{10} + u^2w'wP_3 - u^2w'u2P_{11} + w^3P_6 \\ &\quad +u^2v'vP_5 - uw'vwP_6 - u^2v'w'P_{10} - u^2w'v'P_8) \\ &\quad /(u^2w'^2 + u^2v'^2 - 2uvw' - 2uwv' + v^2 + w^2). \end{aligned} \tag{5.16}$$

After applying this formula three times in a macrotriangle (for every subtriangle together with the adjacent subtriangle of the neighbour macrotriangle), the elevations of the four inner control points (P_{13}, P_{14}, P_{15} and P_{16} in figure 5.5) have to be recalculated to provide G_2-continuity within the macrotriangle. Figure 5.9 shows a perspective view of the test dataset 'Zuestoll' interpolated with the smoothed version of the Clough-Tocher interpolant.

5.6 Considering breaklines in triangular Clough-Tocher Bézier surfaces

In Chapter 4, it was stated that it is often desirable to abandon G_1-continuity at selected locations to represent sharp linear features, such as ridges, lake boundaries or road borders. Known breaklines can not only be considered in Coons surfaces but also in triangular Clough-Tocher Bézier surfaces. However, as in the case of the Coons interpolator, unintentional breaklines are the price for the insertion of the intended breakline. Because of the Clough-Tocher split, two possibilities for considering breaklines exist. The first one has the unintentional breaklines at the same locations as the triangular Coons patch with breakline extension. The second one has the unintentional breaklines within the macrotriangles, which have an edge in common with a breakline. If the Clough-Tocher interpolant is smoothed using the method described in section 5.5, care should be taken to omit smoothing over breaklines.

5.6.1 Unintentional breaklines at the edges of macrotriangles

This possibility of inserting breaklines is similar to the one explained in Chapter 4 for the Coons patch. Unintentional breaklines occur at the same locations and only at the edges of macrotriangles. Figure 5.10 shows three macrotriangles adjacent to a breakline. Three types of normal vectors are used:

- *ordinary normals,* which are estimated by considering all the triangles surrounding the data point.

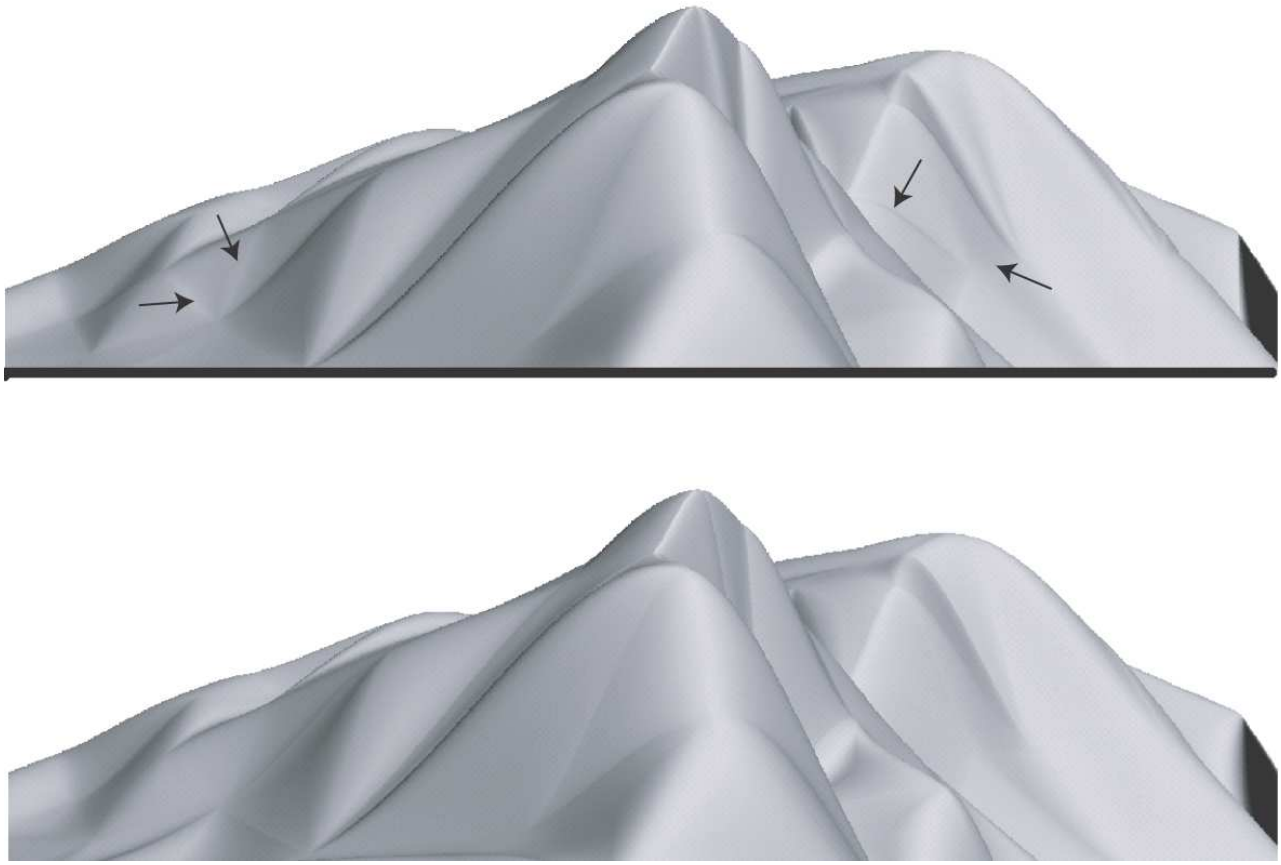

Figure 5.9: Test dataset 'Zuestoll' interpolated with linear interpolated cross-derivatives (above) and with smoothed Clough-Tocher Bézier splines (lower picture). Some artifacts (lines with similar color along triangle edges) occurring when using linear interpolated cross-derivatives are marked with arrows.

- *partial normals* for a data point are valid only for one side of the breakline. The estimation only considers the region on this side of the breakline.
- *intermediate normals* for a data point are valid only for one triangle. It is calculated as the normal to the plane, which contains the connections between the data point and the two adjacent control points. In Figure 5.10, the intermediate normal at P_1 for triangle $P_1P_2P_3$ is the vector orthogonal to $\overline{P_1cp_1}$ and $\overline{P_1cp_6}$.

As a first step, the ordinary normals of the data points belonging to the macrotriangles under consideration are estimated. The control points on the edges of those macrotriangles are calculated such that each is on the plane passing through the next data point and orthogonal to the ordinary normal. Then, for each data point on a breakline, the partial normals have to be estimated. On each triangle edge connecting a data point on a breakline with one that is not on a breakline, the elevation of the control point next to the point on the breakline has to be recalculated such that it is on the

Figure 5.10: Test dataset ’Zuestoll‘ interpolated with Clough-Tocher Bézier splines and breakline extension. The unintentional breaklines are within the macrotriangles (Section 5.6.2). Arrows show some locations of unintentional breaklines

plane orthogonal to the partial normal. Then, for the calculation of the remaining control points within the macrotriangle, the intermediate normal has to be used at the data point on the breakline. Using the ordinary normals (for points, which are not on the breakline) and intermediate normals (for points, which are on the breakline) the calculation of these remaining control points is done exactly the same way as without breaklines. Figure 5.10 shows, why the unintentional breaklines appear at the edges of the macrotriangles. The intermediate normal at P_1 for triangle $P_1P_2P_3$ is not the same as the intermediate normal at P_1 for triangle $P_1P_3P_4$, but the line from P_1 to cp_6 is in both planes. cp_6 is in the plane orthogonal to the partial normal, but cp_3 and cp_1 are not. Therefore, the surface is G_0-continuous across the boundary curve from P_1 to P_3, but not G_1-continuous.

5.6.2 Unintentional breaklines within macrotriangles

For this method, the control points on the boundary curves are estimated first using the ordinary normals at the data points. Then, the partial normals are estimated for the data points on the breakline. For the boundary curves, which connect a breakline point with a non breakline point, the elevation of the control point next to the breakline point is recalculated using the partial normal of the breakline point. For the calculation of the remaining control points, the partial normals at the breakline points and the ordinary normals at the non breakline points are used. Because of this, the unintentional breaklines are within the macrotriangles. In triangle $P_1P_2P_3$ in Figure 5.10 e. g., P_1, cp_3 and cp_6 are in the plane orthogonal to the partial normal at P_1. The elevation of cp_1 in contrast has been estimated using the ordinary normal at P_1. Therefore, an unintentional breakline is present from P_1 to cp_{10}, but not at the boundary curve from P_1 to P_3. This situation is different from the one described in 5.6.1, where P_1, cp_1, cp_3 and cp_6 are in the plane defined by P_1 and the intermediate normal.

5.6.3 Comparison of the two approaches

Figure 5.12 shows both an intended breakline and the locations where unintentional breaklines appear, using the two approaches described. The question as to which of these approaches is superior

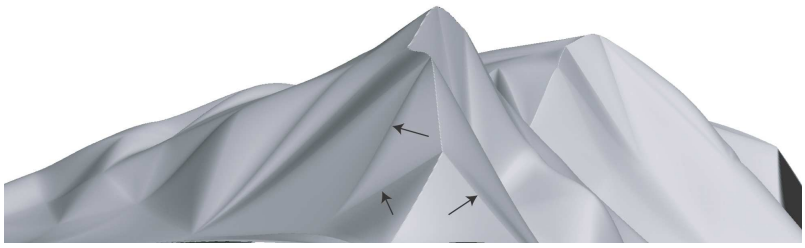

Figure 5.11: Test dataset ’Zuestoll‘ interpolated with Clough-Tocher Bézier splines and breakline extension. The unintentional breaklines are at the edges of macrotriangles (Section 5.6.1). Arrows show some locations of unintentional breaklines.

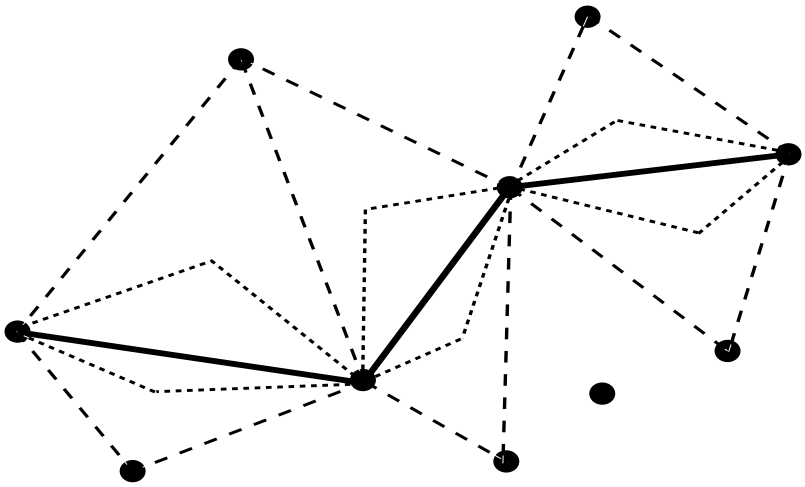

Figure 5.12: Locations where unintentional breaklines occur. Black circles are data points and the bold line represents an intended breakline. Dashed lines are unintentional breaklines at boundary curves of the triangulation, which are present when using the approach described in section 5.6.1. The doted lines are unintentional breaklines within macrotriangles, as they appear when using the approach of section 5.6.2.

cannot clearly be answered. The number of unintentional breakline segments is smaller using the approach which places the unintended breaklines at the edges of the macrotriangles. If the unintentional breakline segments are within the macrotriangles, the summed length of the unintentional segments is lower compared to the second approach. However, it can be argued that the unintentional breaklines therefore form sharper edges than when using the first approach.

Chapter 6

Mesh refinement with the Ruppert algorithm

6.1 Introduction

The more irregularly distributed data points are, the more frequent are long and thin triangles, even if the Delaunay criterion is used for triangulation. In a constrained triangulation, the number of triangles with small angles is even higher. Such triangles with small angles may cause undulations of cubic surfaces (Figure 6.1) as a triangular surface patch has to match the cross-derivatives on all edges of the triangles. If one angle in a triangle is very small, the change of derivatives between two edges needs to happen within a small groundplan distance. The tendency to undulation depends also on the smoothing constraints. A surface, which is forced to change its derivatives slowly is more likely to undulate. Therefore the undulations are more accentuated if the smoothed Clough-Tocher method is used instead of the one with linearly interpolated cross-derivatives, because the curvature change across the triangle border is stronger with linearly interpolated cross-derivatives. Using G_2-continuous elements would increase the undulations.

A possible strategy to avoid triangles with small angles is to insert additional points into the triangulation such that the small angles disappear. Such points are referred to in the literature as Steiner points (Bern and Eppstein, 1995). Two classes of methods exist for point insertion: quadtree based methods and variations of the Ruppert algorithm. Because the quadtree based methods usually insert more Steiner points to remove the small triangles, the Ruppert algorithm is considered in this chapter.

6.2 The Ruppert algorithm

Given a constrained Delaunay triangulation, the Ruppert algorithm uses a sequence of two low-level operations to eliminate triangles with small angles. In the first operation, forced segments are split by inserting a point in the middle of the segments. The second operation inserts points in the circumcenters of triangles with a small angles (Miller et al., 2003).

In this way, the algorithm is able to eliminate all angles below a specified threshold. The threshold can be chosen to be up to about 20°. If the value is too high, it is possible that the algorithm never terminates. If two forced segments join in an angle below the threshold, the algorithm cannot eliminate this angle.

Figure 6.1: Long and thin triangles strongly accentuate undulation of cubic surfaces (indicated by arrows). The picture shows a surface interpolated with the smoothed Clough-Tocher Bézier method described in section 5.5.

Different versions of when to carry out these two operations exist. The one implemented in the prototype software for this thesis is nearly the same as the original method described in Ruppert (1993). The only exception is that the implemented method preserves the convex hull by treating hull edges as forced edges. Ruppert's original algorithm in contrast starts with a set of points and forced segments, adds the four corner points of the bounding box and applies a constrained Delaunay triangulation.

To use the same terminology as Ruppert (1993), a point is *encroached* upon a segment if it is inside the diametral circle through the endpoints of the segment. The first step of the algorithm is to split all forced edges which are encroached by a point. If a split edge is encroached by a point, it is split again, until no point encroaches upon a forced edge.

Then, the triangle containing the smallest angle below the threshold is found and the position of the circumcenter calculated. If the circumcenter does not encroach upon any forced segments, it is added to the triangulation. Otherwise, the segments upon which the point is encroached are split. This procedure is repeated until no angle is below the specified threshold or the angles below the threshold cannot be eliminated (e. g. because they are enclosed between two constrained edges).

Figure 6.2 shows an example of how this algorithm works. Part a shows the initial constrained Delaunay triangulation. Dashed lines are forced edges and solid lines normal triangle edges. In part b, point 1 is inserted in the middle of a segment because there was a point encroaching upon it. In part c, point 2 is inserted at a circumcenter of a triangle containing a small angle. The next circumcenter of a small angled triangle is marked with an empty circle in part d of the figure, because it would encroach upon a segment of the convex hull. Therefore, point 3 is inserted instead to split the segment. Point 4 is inserted at a circumcenter of a triangle containing a small angle and point 5 is inserted to split a segment of the convex hull.

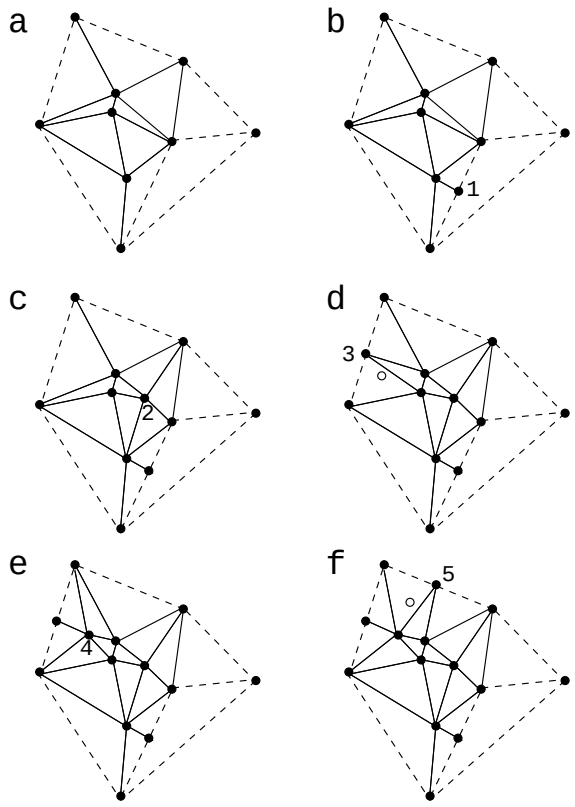

Figure 6.2: Parts a-f show how the Ruppert algorithm inserts points to eliminate small angled triangles.

6.3 Issues related to terrain modeling

In the context of digital terrain modeling the question arises as to how the elevation of the inserted points should be calculated. The Steiner points clearly do not contain new information about the shape of the terrain surface. Their only purpose is to remove small angles. Therefore, it is reasonable to apply the elevation of the digital surface at the position of a Steiner point prior to its insertion. This approach has been implemented in the software prototype. There is no problem with this approach when splitting segments. However, if the circumcenter of a triangle with a small edge is itself in a long and thin triangle, the according surface patch may be inadequate and unsuited for the elevation of the new point. Possible solutions would be to use linear interpolation in such triangles or to use linear interpolation in case the difference between original interpolation and linear interpolation exceeds a threshold value.

6.4 Example

The implementation of the Ruppert algorithm has been applied to the test dataset Albis (near Zurich, Switzerland). The dataset consists of 17’056 input data points from digitised contours and peaks. A Delaunay triangulation constrained to the contour segments was built and a subset of this triangulation is shown in Figure 6.3. Figure 6.5 shows a perspective view of a smoothed Clough-Tocher Bézier surface built from this triangulation and Figure 6.7 depicts a perspective view of a surface interpolated

with Clough-Tocher Bézier splines with linearly interpolated cross-derivatives. Figure 6.5 exhibits undulations in the plain, which are due to triangles with small angles and due to the smoothed Clough-Tocher triangles. The surface with linearly interpolated cross-derivatives in Figure 6.7 does not show this kind of artifacts.

The Ruppert algorithm with a threshold of 17°has been applied to the constrained Delaunay triangulation, resulting in the insertion of 11'236 new points (Figure 6.4). The threshold of 17°has found to be a reasonable compromise between shape and data volume for the test dataset. The elevations of the additional points have been calculated using smoothed Clough-Tocher Bézier splines. As there are no triangles remaining with angles under 17°in this dataset, the undulations of the smoothed Clough-Tocher Bézier splines have disappeared (Figure 6.6).

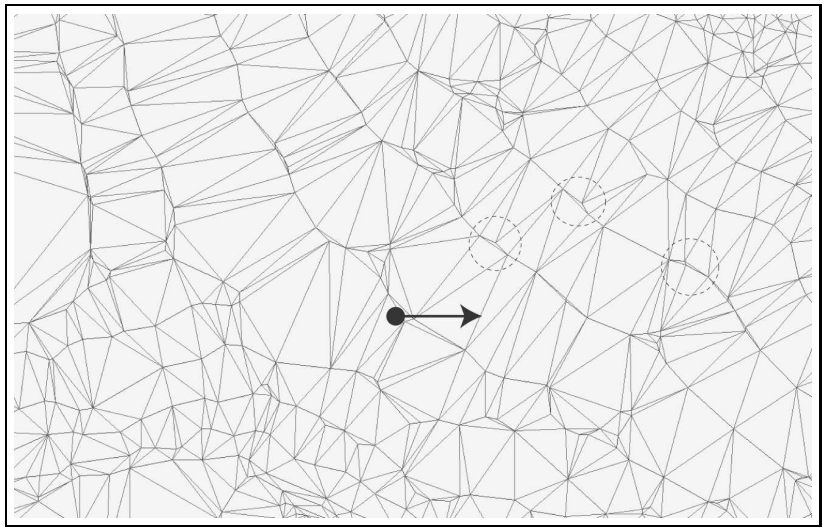

Figure 6.3: Subset of the triangulation of the test dataset Albis without using the Ruppert algorithm. The viewpoint of Figures 6.1, 6.5 and 6.7 is marked with a filled circle. The artifacts visible in Figure 6.1 are marked with transparent dashed circles.

6.5 Discussion

The Ruppert algorithm eliminates triangles with small angles. However, the data volume may grow considerably. How much it grows depends fundamentally on the initial triangulation and the defined threshold angle. In terrain modeling, data configurations which require many Steiner points typically occur if the data points are very irregularly distributed and if many constrained lines are present, for instance, if data from digitised contours are used.

It is important to note that, even if the Ruppert algorithm increases the data point density it does not alter the scale of a model, because the inserted points are interpolated from the ones already present in the data set. Therefore, the additional points do not add any terrain information to the surface model.

The example in Section 6.4 showed, that undulations of the smoothed Clough-Tocher method can be removed by the Ruppert algorithm. However, such undulation can also be weakened by using linearly interpolated cross-derivatives. Therefore the Ruppert method seems to be of special interest for smooth surfaces like the smoothed Clough-Tocher method and even more so for G_2-continuous surfaces.

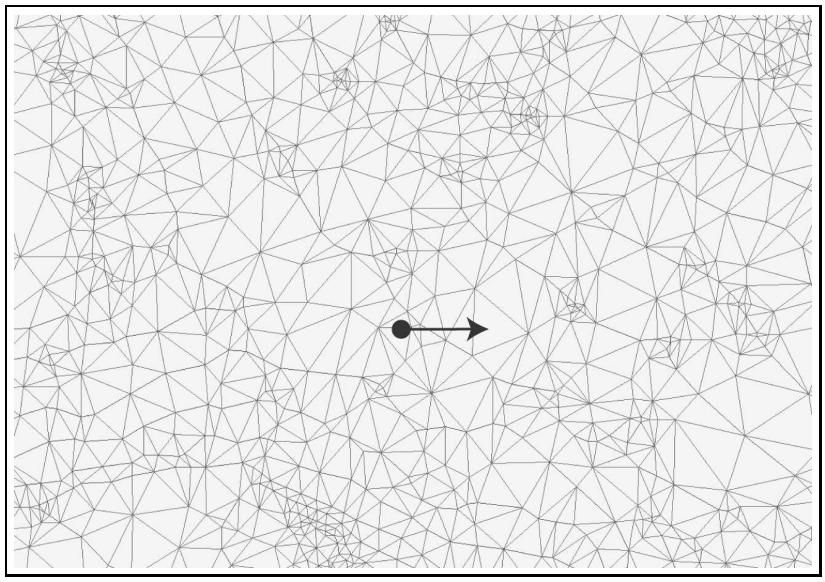

Figure 6.4: Subset of the triangulation of the test dataset Albis with the Ruppert algorithm. The viewpoint of Figure 6.6 is marked with a filled circle. The threshold for small angles is 17°.

Figure 6.5: Perspective view to the test dataset Albis, interpolated with smoothed Clough-Tocher Bézier splines over a constrained Delaunay triangulation.

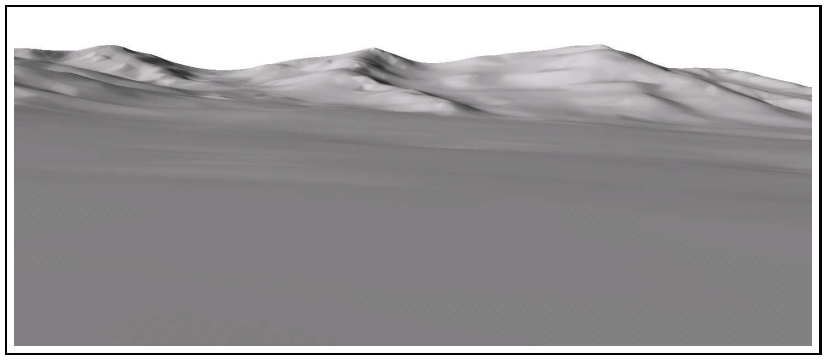

Figure 6.6: Perspective view of the test dataset Albis, interpolated with smoothed Clough-Tocher Bézier splines over a triangulation refined with the Ruppert algorithm with a threshold of 17°.

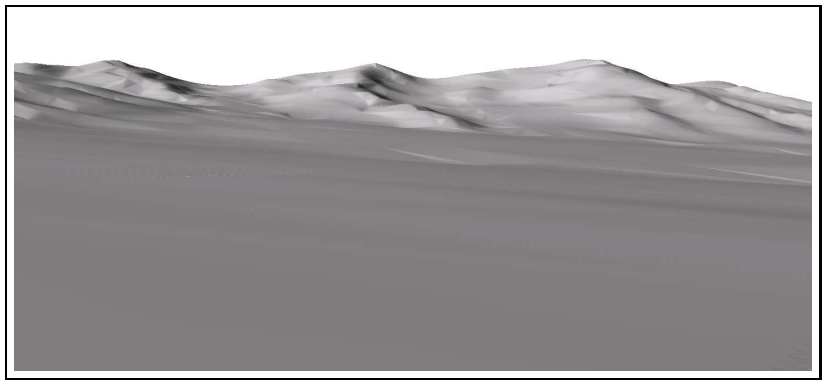

Figure 6.7: Perspective view of the test dataset Albis, interpolated with Clough-Tocher Bézier splines with linearly interpolated cross-derivatives using a constrained Delaunay triangulation.

Chapter 7

Implementation

7.1 Introduction

As a proof of concept, the methods described in this thesis have been implemented in a software prototype called 'tritemo' (*tri*angle based *te*rrain *mo*deller). The program is available under the GPL license and it is thus possible to read and change the source code. It is written in C++ and requires OpenGL and the qt library. The software prototype provides the following features:

- Elevation points, structure lines and contour lines can be read from dxf-files and triangulated using a constrained Delaunay triangulation.
- Zooming and panning have been implemented to explore the triangle mesh.
- Linear interpolation, Coons patches, Clough-Tocher Bézier splines or smoothed Clough-Tocher Bézier splines can be used to create 3D surface views and elevation ascii-grids.
- The triangulation can be interactively edited by swapping edges and inserting Steiner points.
- Triangle edges can be automatically swapped to prevent horizontal triangles.
- Mesh refinement using the Ruppert algorithm is provided.

7.2 Interpolators

Tritemo has an abstract base class `TriangleInterpolator` from which all concrete interpolators are derived (Figure 7.1). TriangleInterpolator provides the method `calcPoint()` to calculate the elevation at any location. The method to calculate the surface normals is `calcNormal()`. By subclassing `TriangleInterpolator`, addition of new triangle based interpolators to the program is straightforward.

At the moment, four concrete interpolation classes are available. `LinTriangleInterpolator` provides a linear interpolation. `CoonsTriangleInterpolator` is a cubic Coons patch with linearly interpolated cross-derivatives as described in Chapter 3. This interpolator considers breaklines as described in Chapter 4. `CloughTocherInterpolator` is a triangular Clough-Tocher Bézier surface with linearly interpolated cross-derivatives (Chapter 5) and `SmoothedCloughTocherInterpolator` an implementation of the smoothed version described in Section 5.5. Both implementations consider breaklines such that the unintentional breaklines are within the macro triangles (section 5.6.2).

All the concrete interpolators have an association to a `Triangulation` object to query the vertex elevations and (for the cubic interpolators) the vertex normals of the triangle containing the interpolation point.

7.2.1 CloughTocherInterpolator

When `CloughTocherInterpolator` is requested to calculate the elevation or the normal at a given point, the interpolator first calls `CloughTocherInterpolator::init()`. There, the associated `NormVecDecorator` object is queried for the vertex coordinates and the vertex normals of the triangle containing the interpolation point as well as for information as to which vertices are on breaklines and which ones are not. Then, with this information, the positions and the elevation of the control points are calculated. To enhance computational performance, the indices of the vertices from the previous run are stored. If the next interpolation point is in the same triangle, the control points do not have to be recalculated. With the control points and the Bernstein polynomials the interpolated elevation or normal is calculated (Chapter 5). `SmoothedCloughTocherInterpolator` works similarly, but additionally, the control points of the three neighbouring triangles have to be calculated in the `init()` method and Lagrange minimisation is performed (section 5.5).

7.2.2 CoonsTriangleInterpolator

CoonsTriangleInterpolator also has a method `CoonsTriangleInterpolator::init()` in which the associated `NormVecDecorator` object is queried for the triangle points and normals of the triangle containing the interpolation point. The coordinates of the vertices and the derivative endpoints are then transformed into the standard triangle. The three boundary curves (in standard triangle coordinates) are stored as objects of type `Bezier3D` (`CoonsTriangleInterpolator::pl1`, `CoonsTriangleInterpolator::pl2`, `CoonsTriangleInterpolator::pl3`). `CoonTriangleInterpolator` has protected methods to calculate the elevations and derivatives of the ruled surfaces; for instance, `CoonsTriangleInterpolator::calcPointRS` calculates the elevation of the ruled surface 1. `CoonsTriangleInterpolator::calcPoint` and `CoonsTriangleInterpolator::calcNormal` then add and subtract the results of the ruled surface and apply the correction term appropriately.

7.3 Tessellation

Tritemo uses the interface `Triangulation` to make the representation of the triangular tessellation exchangeable. At the moment, `DualEdgeTriangulation` is the class maintaining the triangulation and it is used directly by the linear interpolator. `NormVecDecorator` adds the functionality of estimating and managing vertex normals and is used by the cubic interpolators. `NormVecDecorator` objects have an association to a `DualEdgeTriangulation` object and delegate tasks to this object whenever possible. The sequence diagram in Figure 7.3 shows the interaction between `NormVecDecorator` and `DualEdgeTriangulation` when a point is inserted into an instance of `NormVecDecorator`. `NormVecDecorator` can also calculate the partial normal of a breakline vertex for an interpolation point (method `NormVecDecorator::calcNormalForPoint()`), which is used by the cubic interpolators for breakline handling. Because of the `Triangulation` interface and the delegation mechanism, it would be possible to replace the `DualEdgeTriangulation` by subclasses of `Triangulation` using other data structures and algorithms. Because of the common interface, this wouldn't affect `NormVecDecorator`.

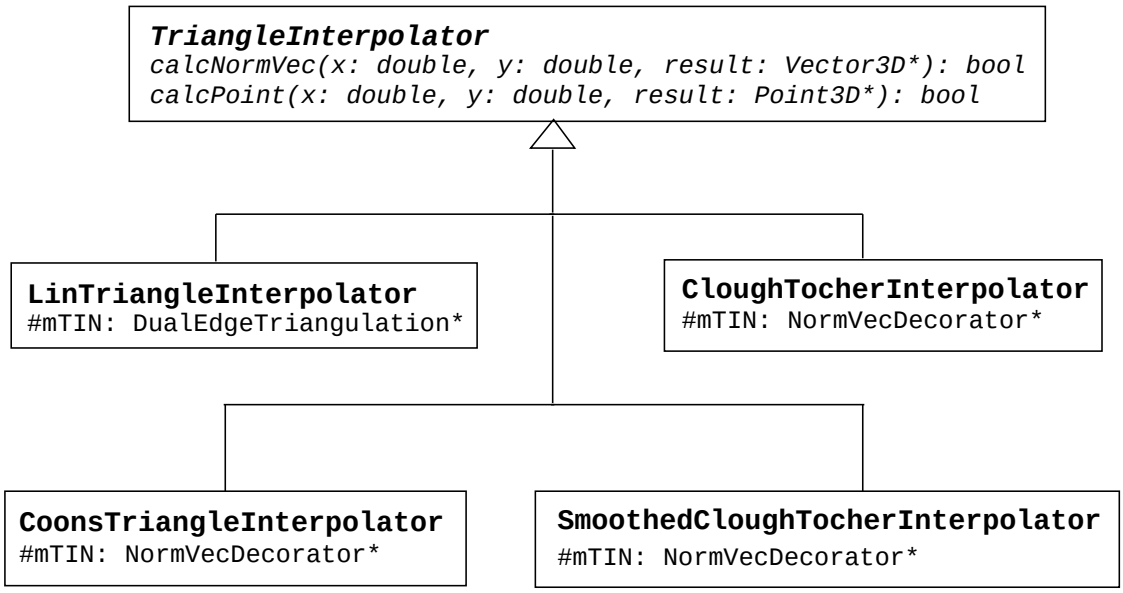

Figure 7.1: UML class diagram of the interface `TriangleInterpolator` and its derived classes

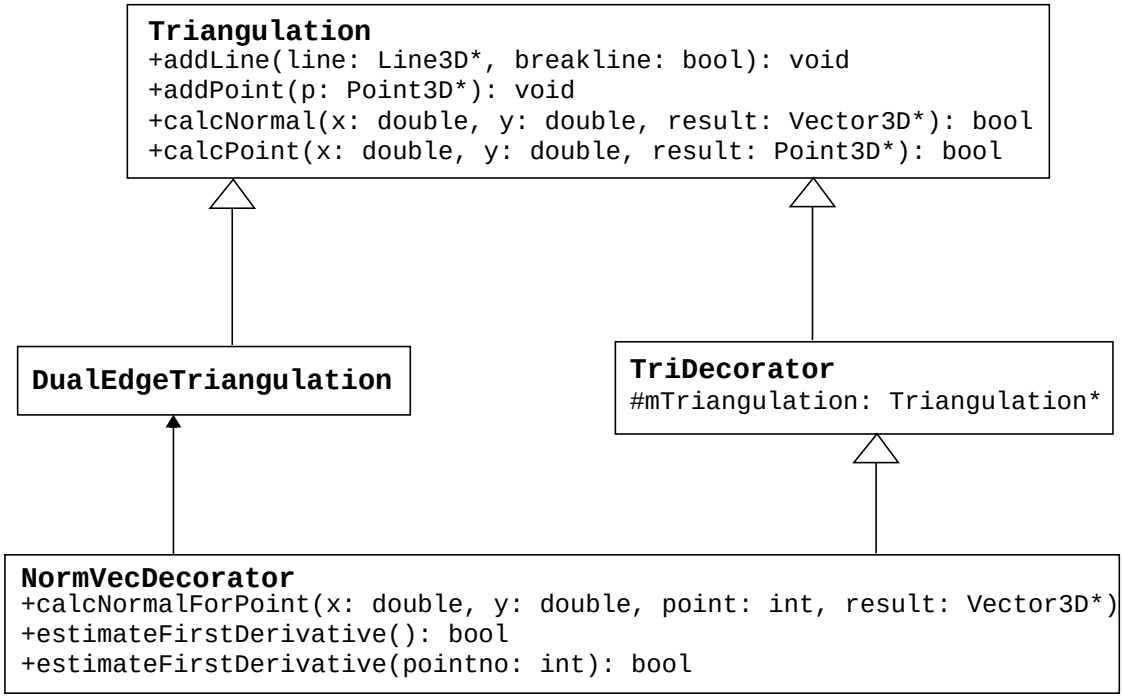

Figure 7.2: `Triangulation` and its derived classes

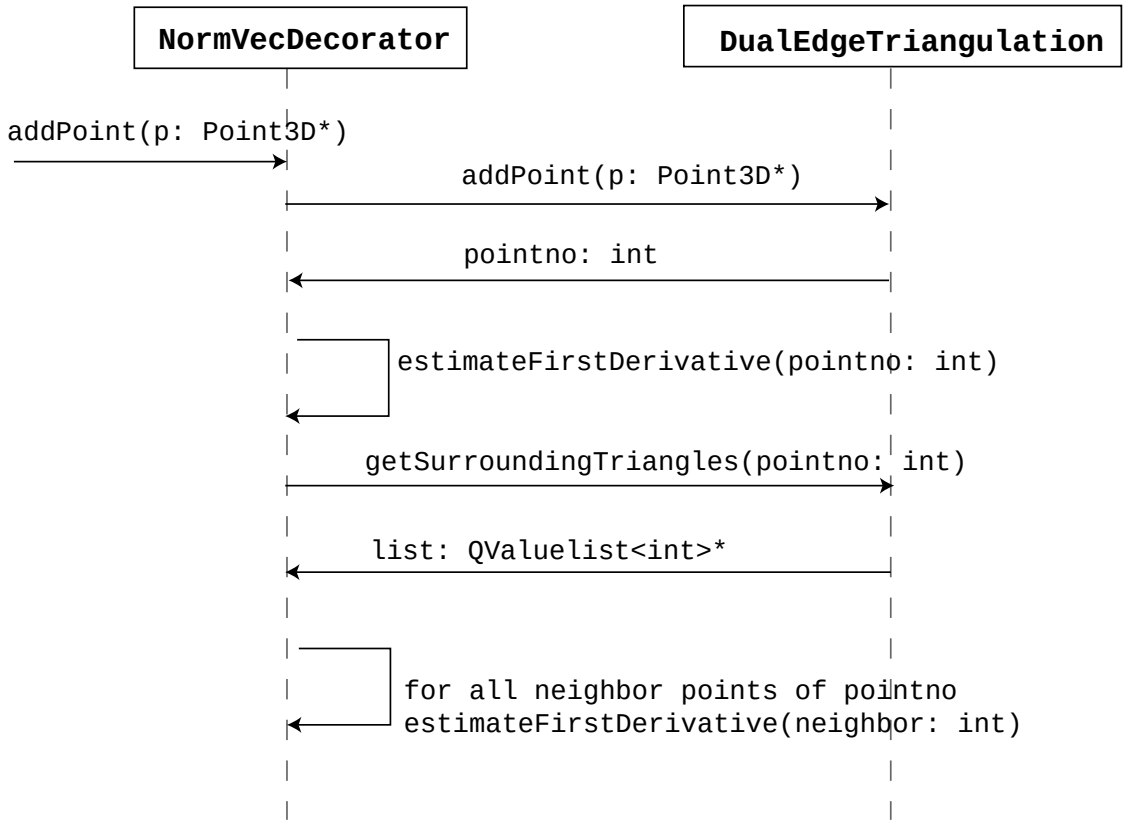

Figure 7.3: Collaboration between `NormVecDecoration` and `DualEdgeTriangulation` when a new point is inserted into NormVecDecorator.

As the name indicates, `DualEdgeTriangulation` uses a dual edge data structure to maintain the triangulation (Heller, 1990; Schneider, 1998). The half edges are represented as objects of type `HalfEdge` and `DualEdgeTriangulation` stores pointers to its half edges in the vector `DualEdgTriangulation::mHal` Each `HalfEdge` stores the indices of its twin edge (that is the edge pointing in the opposite direction) of the vertex it points to and of the next edge in the same triangle (Figure 7.4). With this information it is possible to traverse the triangulation and do operations such as searching the triangle containing an interpolation point or searching the neighbour vertices of a given vertex. HalfEdge further has two boolean flags which maintain the information whether the `HalfEdge` represents a forced edge (HalfEdge::mForced) and whether the HalfEdge belongs to a breakline (HalfEdge::mBreak).

The triangulation is built using an incremental algorithm for constrained Delaunay triangulating (de Floriani and Puppo, 1992). After each insertion, `DualEdgeTriangulation` manipulates the triangulation such that it is a constrained Delaunay triangulation. If a new point is inside the convex hull, this is done as follows using the dual edge data structure. The new point is connected to the vertices of the triangle which contain the new point. Then, if another vertex is inside the circle through a new triangle, the outer edge of this new triangle is swapped (unless this edge is a forced edge and therefore cannot be swapped). After each swap, the triangles connected to the newly inserted point are tested until no vertex is inside the circle through a triangle connected to the new point. Figure 7.5 illustrates this process.

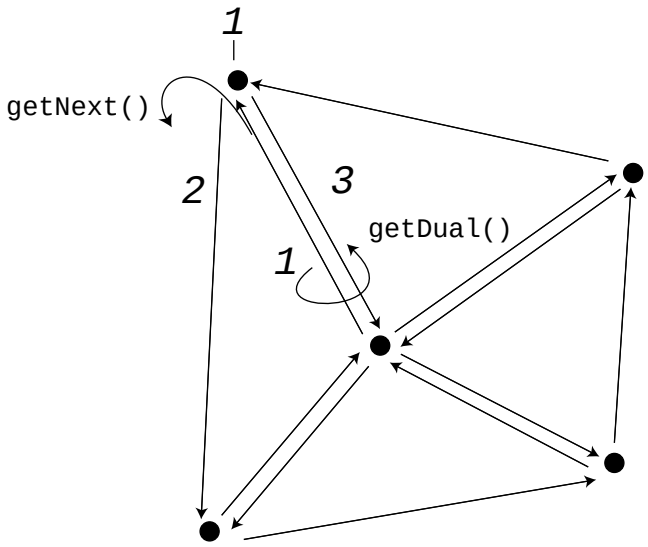

Figure 7.4: DualEdge data structure. Each half edge has pointers to the indices of the dual edge, of its endpoint and of the next edge in the same triangle. For instance, `mHalfEdge[1]->getDual()` returns 3, `mHalfEdge[1]->getNext()` returns 2 and `mHalfEdge[1]->getPoint()` returns 1.

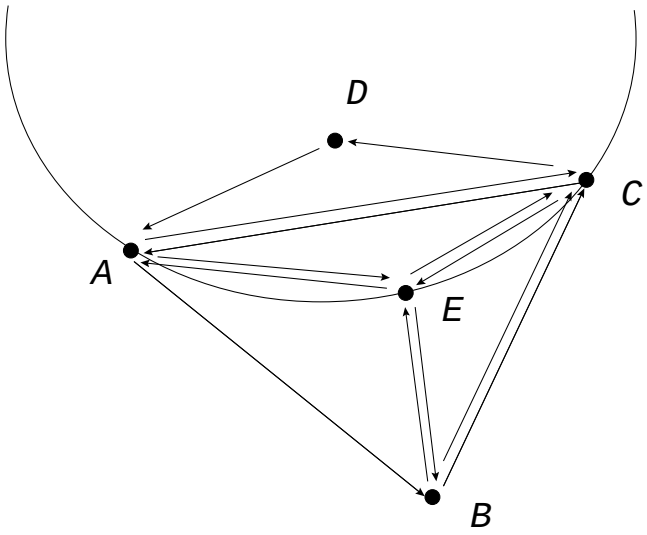

Figure 7.5: Insertion of the new point *E* into `DualEdgeTriangulation`. *E* is connected to *A*, *B* and *C*. Because point *D* is inside the circle through *A*, *D*, *C*, at least one edge swap has to be done do satisfy the Delaunay criterion.

Chapter 8

Evaluation

8.1 Introduction

In this chapter, the methods described in this thesis are evaluated. For this purpose, a set of field data measured with a geodimeter and two sets of artificial data are used. In Section 8.2, an overview of the evaluation approaches is given. Section 8.3 explains the details about how the evaluation is done, Section 8.4 presents the results and Section 8.5 provides a comparative discussion.

8.2 Previous approaches in DEM/ DTM evaluation

There are various approaches for evaluation of DEMs and DTMs, and they can be classified into three broad groups:

- visual inspection of the interpolated terrain models
- quantitative comparison with artificial surfaces
- quantitative comparison with values of known higher precision

8.2.1 Visual inspection

Evaluation of terrain models by visual inspection is a method often cited in the GIS literature. Wood and Fisher (1993) listed a range of mapping techniques to perform this task: 2D raster rendering, pseudo-3D projection, aspatial representations, shaded relief maps, slope and aspect maps and mapping of local detectors. Often, isolines are used to visualise the surfaces (e. g. Akima (1978), Ebner et al. (1980)). In computer aided graphic design (CAGD) reflection lines are used to detect small irregularities in surfaces. A reflection line is the collection of all points which reflect the (parallel) rays of an infinitely distant light source to the direction of an (infinitely far) observer (Farin, 1985). Mitas and Mitasova (1999) and Schneider (1998) detect artifacts in continuous surfaces with the help of a shaded pseudo-3D projection.

Visual inspection has the advantage that it is straightforward for a human to recognise spatial patterns. Artifacts can thus be recognised quickly, whereas in global summary statistics this is not easily possible (Wood and Fisher, 1993).

However, a disadvantage of this approach is that, in the absence of severe artifacts, it is hard to assess how well the surface model represents the properties of a terrain surface. Two different surfaces

may both have a realistic (or plausible) appearance even if they are quite different. A surface may appear visually pleasing while being unsuitable for other applications beside visualisation (Schneider, 2001b).

8.2.2 Comparison with artificial surfaces

A mathematical function with an analytical solution can be used to quantitatively evaluate interpolators. Carter (1992) used an artificial surface to show the effect of data precision on terrain derivatives in gridded terrain models. Corripio (2003) used an expression containing several trigonometric functions to produce a surface with a highly variable relief to compare two methods for calculating slope in grids.

Artificial surfaces provide a means to quickly collect many samples. Knowing the true value of elevation as well as terrain derivatives at every point makes this an attractive technique to create error statistics for comparing interpolation techniques. The possibility to analytically obtain surface derivatives, which cannot be measured clearly in the field, is a major advantage of this approach. It is straightforward to compare not only elevation but also slope, aspect and curvature. However, it can be argued that a mathematical function may not necessarily have the same properties as a real terrain surface.

8.2.3 Comparison with field data of higher precision and accuracy

Analysis on a real terrain can be done by comparing the interpolated values or the terrain derivatives with values of higher precision and accuracy. Skidmore (1989) and Wise (1997) assess the quality of terrain derivatives by comparing the values calculated with gridded DEMs with values measured from topographic maps. Bolstad and Stowe (1994) use fixed points on maps as well as values measured with GPS to evaluate the quality of DEMs. For slope and aspect, field measurements have been used. Giles and Franklin (1996) used field measured values of elevation, slope and profile curvature for quality assessment.

Comparison with values of higher precision and accuracy provide a ground truth for terrain modeling. The field work is often time consuming and needs thorough planning. It is often hard to separate the effect of the varying parameter (e. g. algorithm, data source, interpolation technique) from other effects (see below).

Comparing and evaluating interpolation methods with field data needs to cope with two major problems:

1. Every application uses a terrain model on a certain scale level. This very scale-specific surface, however, does not exist in reality and can therefore not be measured (Schneider, 2001a).

2. There are also factors other than the interpolation method which influence the result of a DTM:

   - the discretisation of the continuous terrain surface into discrete data points;
   - the triangular tessellation (different triangulations lead to different results); and
   - errors in the elevation data and in the reference data.

   To evaluate interpolation methods, the uncertainty introduced through interpolation has to be assessed as large compared with the uncertainties introduced through discretisation, tessellation and base data. One of the examples here will show what happens if the errors introduced through the elevation data have about the same magnitude as the influence of the different interpolators.

For this field study this means the following:

- The features not represented in the elevation data have to be small. In practice, no dataset contains all features which are of interest for a particular application. This, however, is a problem of the input dataset and not of the interpolation method used on this dataset. The idea of an interpolation method is to specify a continuous surface using all the information contained in a dataset. Information which is not (implicitly or explicitly) contained in a dataset cannot be introduced by interpolation. If the features not included in the dataset are too large, comparison with field values will rather reflect the quality of the data set than the quality of the interpolation methods.
- The triangular tessellation must portray the terrain well (no edges crossing valleys and ridges, no triangles which are extremely long and thin).
- The elevation values used to create the triangular terrain models have to be accurate and precise.

8.3 Methodology

8.3.1 Creation of the triangular tessellations

Artificial surfaces

The first artificial surface (Figure 8.1) has been created using the following function:

$$f(x,y)[m] = 500 + (sin(\frac{2\pi}{200}x) + sin(\frac{2\pi}{200}y))100. \tag{8.1}$$

This function is evaluated in the range $x = [100m, 300m]$ and $y = [100m, 300m]$. 162 data points have been selected to build a tessellation for the interpolation methods.

The second artificial function (Figure 8.2) includes a breakline and has been generated in the domain $x = [-100m, 100m]$, $y = [-100m, 100m]$ using the function

$$f(x,y)[m] = max(300 - 0.04x^2 - 0.04y^2, 200). \tag{8.2}$$

The shape of this surface is such that there is a peak at (0/0) and a circular breakline with elevation 200 and radius 100. In order to create a triangular tessellation, 200 points have been selected randomly and 50 points on the break circle have been used.

Comparison with field data of higher precision and accuracy

The test area is a square of 200m*200m near Menzingen (Zug, Switzerland) in a landscape formed by glacial deposition. It contains a hill, a small road and a plain (Figure 8.3). To build a triangular tessellation, initially a dataset captured by analytical photogrammetry has been used (Digitales Terrainmodell Kanton Zug). However, during a first evaluation, we found that the photogrammetrically captured terrain data was systematically 20-30cm below the values measured with the geodimeter. An exact comparison is difficult as the data points of the two datasets are not the identical. This example shows what happens if the uncertainties of the data values are not considerably smaller than those of the interpolation. The result of this evaluation was that the interpolator which yielded the lowest elevation value was in most cases the one closest to the field measured value.

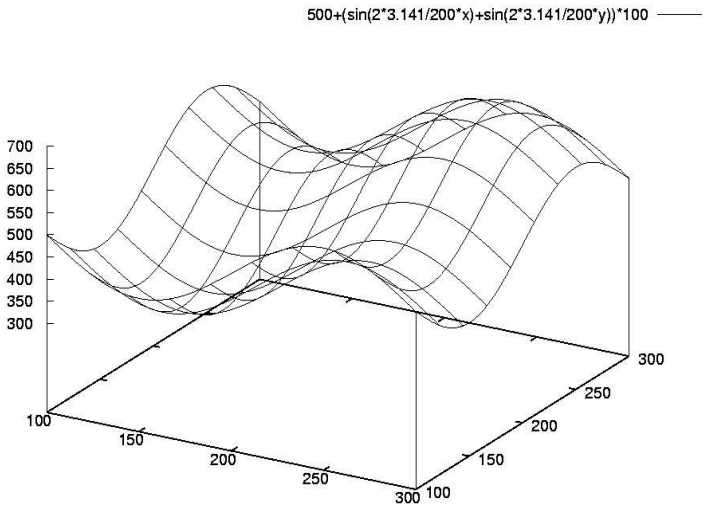

Figure 8.1: First artificial function.

Figure 8.2: Second artificial function.

To cope with this problem, another 230 data points were measured by geodimeter at about the same locations as the data points of the photogrammetrically captured data. This includes two breaklines at the sides of the road (as it was the case in the photogrammetric data set). With this data, a constrained Delaunay triangulation has been built using the algorithm described in de Floriani and Puppo (1992).

In order to test whether the results of the study are stable two new datasets have been created for the triangulation by swapping 40 (respectively 80) data points from the triangulation to the control data set and 40 (respectively 80) back.

Figure 8.3: Test area viewed from the north-east direction.

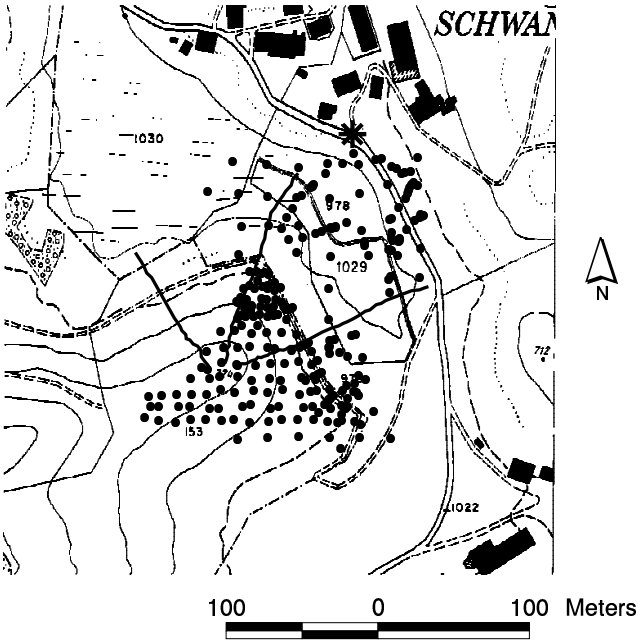

Figure 8.4: Locations of the sample points and of the profiles. The viewpoint of the photo in figure 8.3 is indicated by a star.

8.3.2 Interpolation methods

Based on the created triangulations, four different interpolators implemented in this thesis have been used to calculate elevation values:

- Linear interpolation (Chapter 2)
- Triangular Coons patch (linearly interpolated cross-derivatives) (Chapter 3)
- Clough-Tocher Bézier spline (linearly interpolated cross-derivatives) (Chapter 5)

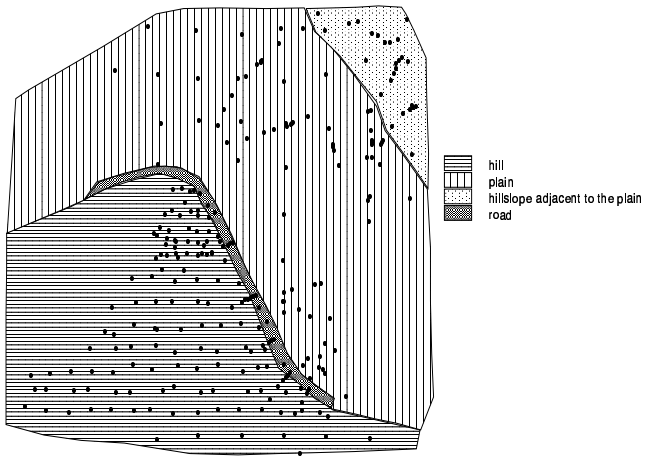

Figure 8.5: Extent of the subareas hill, plain, hillslope adjacent to the plain and road.

- A smoothed version of the Clough-Tocher spline (Chapter 5)

8.3.3 Comparison techniques

Visual inspection

The interpolators are compared by four shaded 3D projections from the same location. These pictures have been inspected to find irregularities and artifacts of the interpolation and the triangular tessellation.

First artificial surface

The values of the interpolators have been compared to those calculated from the function at 25922 points which are located on a grid. Then, several analyses have been made. The following outcomes are presented here:

- The mean (unsigned) deviations of the four interpolation methods to the artificial surface have been calculated and compared.
- The (signed) deviations of the interpolators have been correlated with the values of total curvature.
- The mean deviations of slope calculated with the interpolation methods and the slope values calculated with the artificial surface have been compared.
- Two diagrams have been produced showing the relation between the slope values calculated with the linear and the Coons interpolation and those calculated from the artificial function.

Second artificial surface

A Coons interpolator with breakline extension and one without consideration of breaklines have been applied to the created tessellation. For quantitative evaluation, 5000 points have been randomly selected. All points have been selected so that they lie within triangles adjacent to breaklines because

Table 8.1: Descriptive statistics for the deviations of the interpolated values from the measured elevations over the whole test area.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	263	263	263	263
mean	0.42m	0.31m	0.31m	0.31m
minimum	0.0007m	0.0007m	0.0013m	0.0018m
maximum	5.24m	5.50m	5.50m	5.51m
standard deviation	0.47m	0.45m	0.45m	0.45m

Table 8.2: Descriptive statistics for the deviations of the interpolated values from the measured elevations on the hill.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	155	155	155	155
mean	0.45m	0.29m	0.29m	0.29m
minimum	0.0074m	0.0007m	0.0013m	0.0028m
maximum	3.80m	3.79m	3.78m	3.79m
standard deviation	0.39m	0.36m	0.36m	0.36m

only these locations are affected by the modeling of breaklines. The interpolated elevations at the points are compared to the values calculated from equation 8.2.

Comparison with field data of higher precision and accuracy

Global statistics The first set of analyses has been made by evaluating the deviations of the interpolated values from the field measurements. In Figure 8.1, some descriptive statistics of these deviations have been computed for the whole test area. Additionally, the same statistics have been computed for the different subareas hill, road, plain and hillslope adjacent to the plain (Figures 8.5 - 8.5). To assess how stable these results are with respect to the triangular tessellation, two new data sets of point and evaluation data have been created. The first data set was created by randomly selecting 40 points from the existing point data set as well as in the existing evaluation data set and by shifting these selected points from one data set to the other. The second data set was created the same way, but with 80 points selected per data set. Global descriptive statistics for the deviations of the different interpolators from these evaluation data sets can be seen in Tables 8.6 and 8.7.

Table 8.3: Descriptive statistics about the deviations of the interpolated values from the measured elevations on the road.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	15	15	15	15
mean	0.22m	0.23m	0.23m	0.23m
minimum	0.0242m	0.0451m	0.0576m	0.0576m
maximum	0.70m	0.69m	0.69m	
standard deviation	0.21m	0.18m	0.17m	0.17m

Table 8.4: Descriptive statistics for the deviations of the interpolated values from the measured elevations on the plain.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	67	67	67	67
mean	0.31m	0.29m	0.29m	0.29m
minimum	0.0007m	0.0171m	0.0191m	0.0018m
maximum	1.05m	0.88m	0.88m	0.87m
standard deviation	0.28m	0.22m	0.22m	0.23m

Table 8.5: Descriptive statistics for the deviations of the interpolated values from the measured elevations on the hillslope adjacent to the plain.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	26	26	26	26
mean	0.68m	0.53m	0.53m	0.54m
minimum	0.0067m	0.0049m	0.0051m	0.0042m
maximum	5.24m	5.50m	5.50m	5.51m
standard deviation	1.01m	1.05m	1.05m	1.05m

Table 8.6: Descriptive statistics for the deviations of the interpolated values from the measured elevations with 80 points randomly changed between the control data set and the base data set.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	262	262	262	262
mean	0.46m	0.33m	0.33m	0.33m
minimum	0.0006m	0.0002m	0.0004m	0.0013m
maximum	12.2m	12.36m	12.36m	12.3543m
standard deviation	0.82m	0.81m	0.81m	0.81m

Table 8.7: Descriptive statistics for the deviations of the interpolated values from the measured elevations with 160 points randomly changed between the control data set and the base data set.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	261	261	261	261
mean	0.64m	0.52m	0.52m	0.53m
minimum	0.0079m	0.0033m	0.0041m	0.0006m
maximum	25.07m	25.27m	25.27m	25.28m
standard deviation	2.30m	2.32m	2.32m	2.32m

Spatial patterns The second set of analyses consists of maps to analyse spatial patterns. In Figure 8.6, the (signed) differences between the values of the linear interpolation and the field measured values are mapped. Figure 8.7 does the same for the values of the Coons interpolator. Figure 8.8 compares the linear interpolator and the Coons interpolator by subtracting the (unsigned) deviation of the Coons interpolation from that of the linear interpolation. Thus, positive values mean that the Coons interpolator is closer to the measured value and negative values indicate that the linear interpolation is closer. The same comparison has been made between the smoothed Clough-Tocher and the Coons interpolator (Figure 8.9).

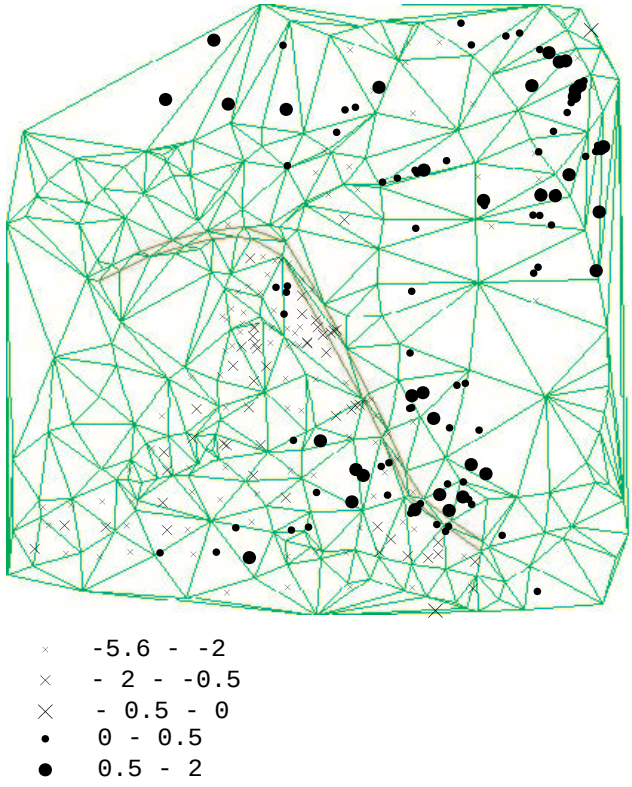

Figure 8.6: Linear interpolation below (negative values) and above (positive values) surface (in meters).

Profiles The three measured profiles have been mapped to 2D-plots to reveal which interpolator is below or above the real surface. Furthermore, the deviation of the interpolators have been magnified by a factor of 10 to ease interpretation (Figures 8.10 -8.12).

Breaklines For all the analyses, the cubic methods have been run with a breakline extension, ensuring that both edges of the road are represented as breaklines in all surfaces. To test whether the breakline extension is useful or not the differences between the Coons method with and without the breakline extension have been analysed. Points which have a different elevation value due to the breakline extension have been selected. Descriptive statistics of the deviation of these points from the measured values have been made (Table 8.12) and a map shows where the breakline extension makes a difference (Figure 8.13).

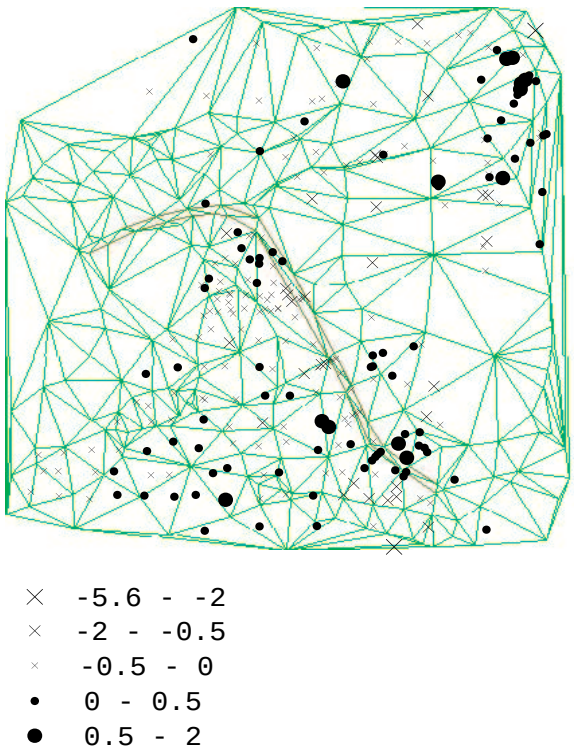

Figure 8.7: Coons interpolation below (negative values) and above (positive values) surface (in meters).

8.4 Results

8.4.1 Visual inspection

Looking at the 3D pictures of the test area (Figures 8.15 - 8.18) the linear interpolation shows, not surprisingly, the most apparent artifacts. The linear facets do not represent the hill well, because most of the breaklines at the triangle edges have no geomorphological meaning. The triangle Coons patch and the Clough-Tocher method (both with linearly interpolated cross-derivatives) appear more realistic because they smooth the surface and model the convexity and concavity of the terrain. Artifacts of the triangular tessellation are clearly visible and considered as an effect of the linearly interpolated

Table 8.8: Descriptive statistics for the deviations of the Coons patch surface with and without breakline extension (only points with different elevations are considered).

	with breakline	without breakline
count	75	75
mean	0.29m	0.36m
minimum	0.0007m	0.0182m
maximum	1.18m	1.20m
standard deviation	0.24m	0.32m

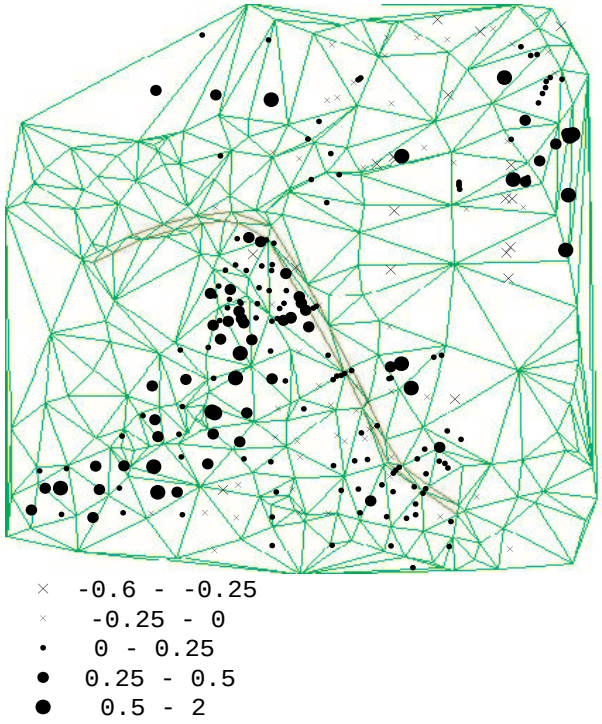

Figure 8.8: Positive value: Coons interpolator is closer to the measured value. Negative value: linear interpolator is closer to the measured value (in meters).

cross-derivatives. The smoothed Clough-Tocher interpolator removes many of these artifacts and thus has a more realistic appearance than the other interpolators.

8.4.2 First artificial surface

The mean deviation of the linear surface from the artificial surface is much higher than that of the other interpolators. Within the cubic interpolators, the smoothed Clough-Tocher interpolation has the lowest mean deviation, but the differences are small (Table 8.9).

Looking at the correlation between curvature and (signed) difference between the interpolators and the function, the differences belonging to the linear interpolation show a high correlation with the curvature of the function (Table 8.10). The correlation coefficients belonging to the cubic interpolators, on the other hand, are close to zero. This means that the linear interpolation strongly tends to underestimate the artificial surface in convex areas and overestimate the surface in concave areas, which is an expected property.

The mean deviations of slope values from the slope values of the function are small for all interpolation methods (Table 8.11). Nevertheless, the deviations of the cubic interpolators are also considerably lower. The diagrams showing the distribution of these deviations for the linear interpolation and for the Coons interpolation (Figures 8.19 and 8.20) reveal that the deviations of slope are lower with increasing values of slope on the artificial surface. The low mean deviations of slope seem therefore to occur due to the fact that the chosen artificial surface is steep in most locations. In Figure

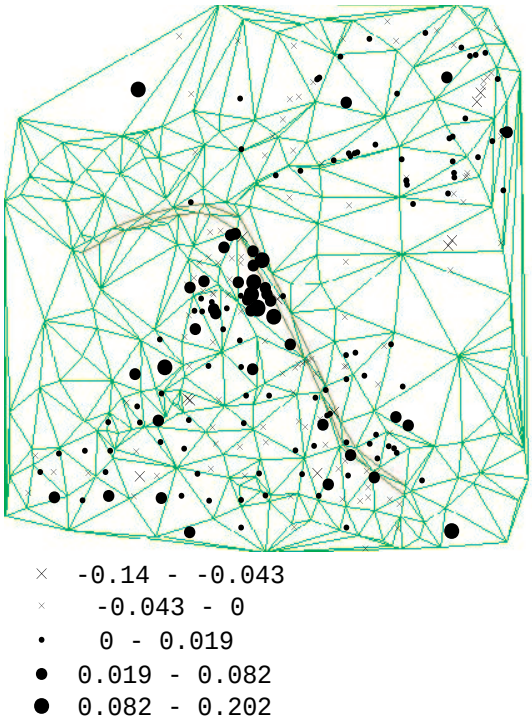

Figure 8.9: Positive value: Coons interpolator is closer to the measured value. Negative value: smoothed Clough-Tocher interpolator is closer to the measured value (in meters).

8.19, points which are on the same triangle facet are well visible since they have the same slope and form stripes in the figure.

8.4.3 Second artificial surface

Table 8.12 shows that using the breakline extension reduces the average error from 3.91 to 1.88 on this artificial dataset, that is, by 51.92%.

Table 8.9: Descriptive statistics for the deviations of the interpolated values from the first artificial surface.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	25921	25921	25921	25921
mean	2.61m	0.87m	0.87m	0.83m
minimum	0.00047m	0.00001m	0.00003m	0.00003m
maximum	10.90m	7.16m	7.17m	7.04m
standard deviation	2.12m	1m	1m	0.99m

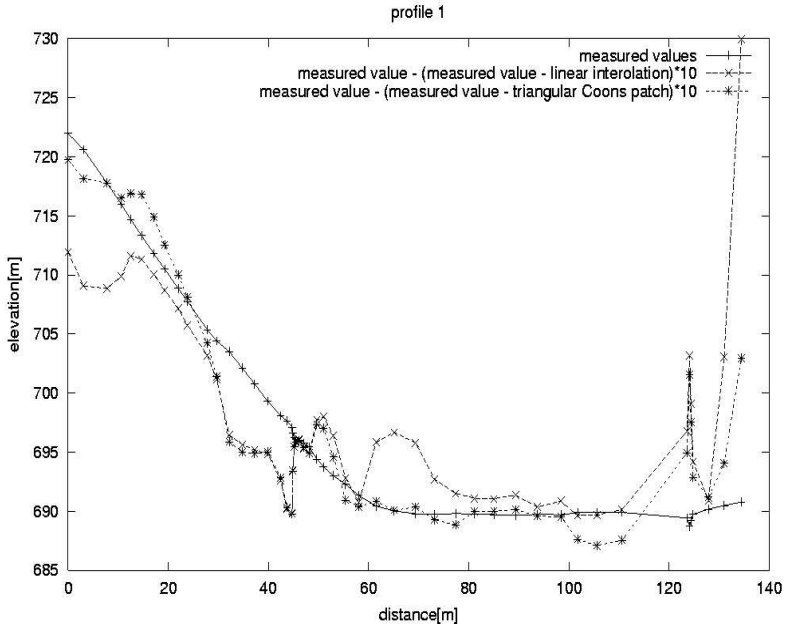

Figure 8.10: Profile 1. The deviations of the interpolators from the measured value have been multiplied by 10 to visualise where they are above and below the terrain surface, respectively.

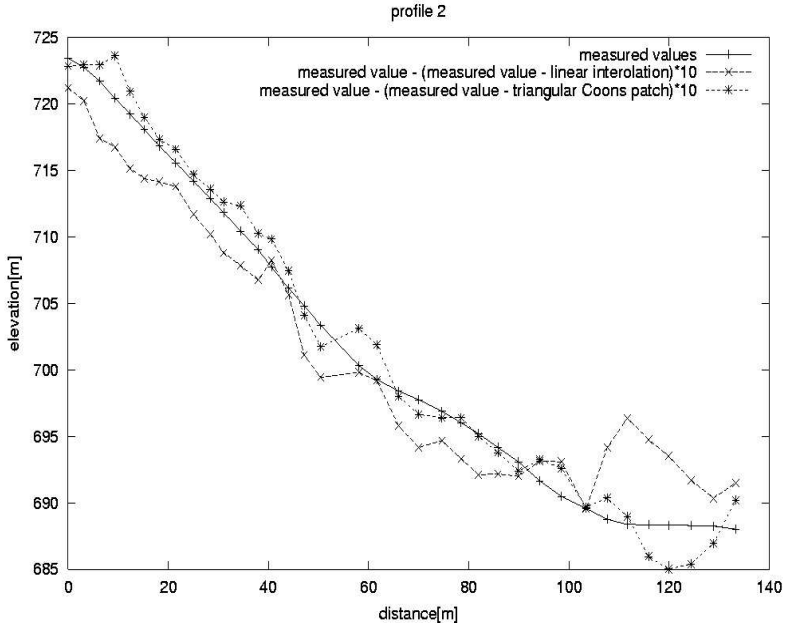

Figure 8.11: Profile 2. The deviations of the interpolators from the measured value have been multiplied by 10 to visualise where they are above and below the terrain surface, respectively.

8.4.4 Comparison with field data of higher precision and accuracy

Comparing the interpolators with the field measured data over the whole area, the mean deviations of the cubic interpolators are considerably lower than that of the linear interpolation (Table 8.1). The

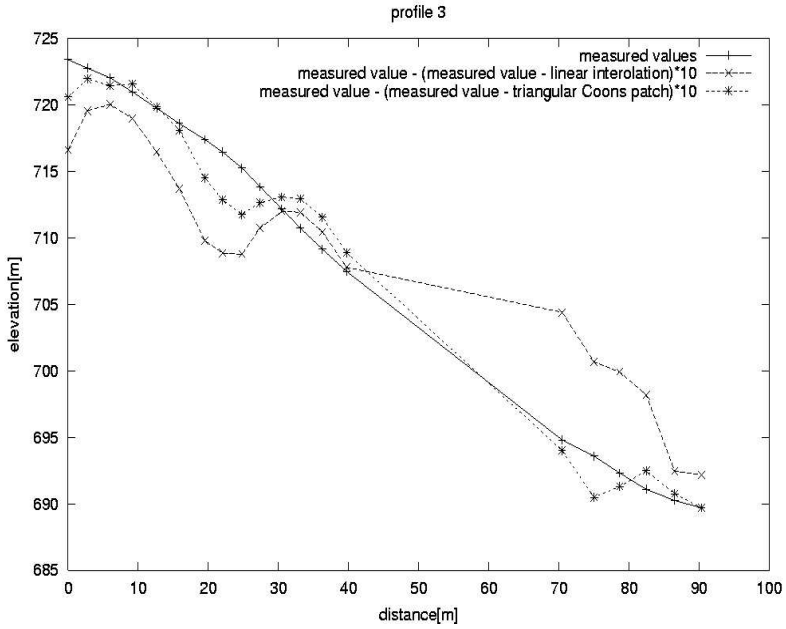

Figure 8.12: Profile 3. The deviations of the interpolators from the measured value have been multiplied by 10 to visualise where they are above and below the terrain surface, respectively.

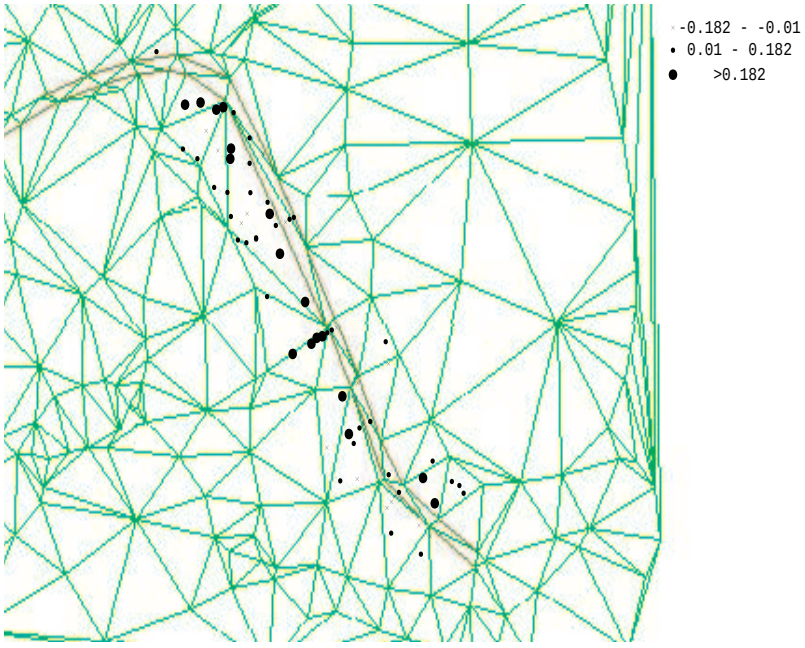

Figure 8.13: Positive values: the surface with breakline extension is closer to the field measured value. Negative values: the surface without breakline extension is closer (in meters).

other statistical measures are similar for all four interpolators. The differences between the three cubic

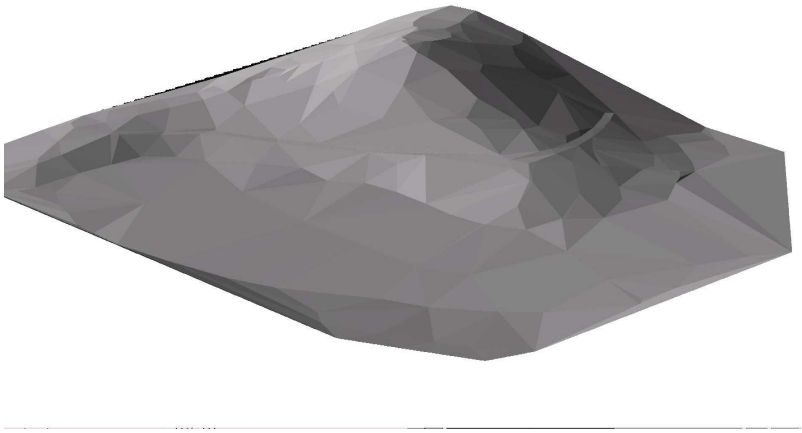

Figure 8.14: Overview of the linearly interpolated test area (view from north-east).

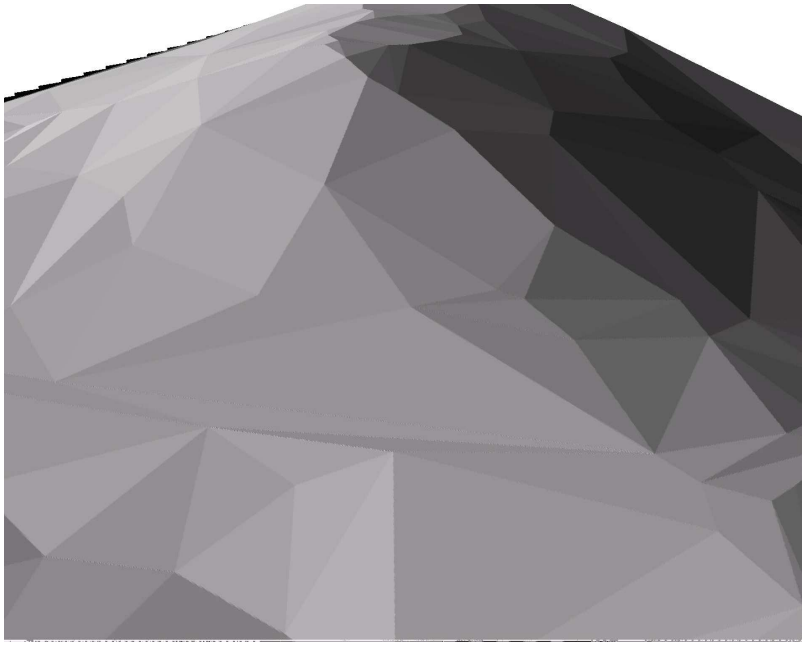

Figure 8.15: Detail of the linearly interpolated test area (view from north-east).

interpolation methods are very small, which can be seen in the tables 8.1 - 8.5as well as in figure 8.9. It can be seen in Tables 8.6 and 8.7 that these observations remain if the triangulation is changed. The maximum deviation rises from approx. 5m (original triangulation) to approx. 12m (80 points

Figure 8.16: Test area interpolated by the triangular Coons patch (view from north-east).

Figure 8.17: Test area interpolated with the Clough-Tocher method using linear cross-derivatives (view from north-east).

swapped) and approx. 25m (160 points swapped). This is because the original triangulation consists of relatively well shaped triangles. The more points are swapped, the worse shaped the triangulation

Figure 8.18: Test area interpolated with the smoothed Clough-Tocher method (view from north-east).

Table 8.10: Correlation between the (signed) differences of the interpolators from the first artificial surface and the total curvature.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
correlation	0.64	0.08	0.08	0.07

becomes, resulting in higher maximum deviation between the measured values and the interpolated ones.

Looking at the results for the different areas, the cubic interpolators have considerably lower mean deviation on the hill and on the hillslope adjacent to the plain. On the road and in the plain, the mean deviations are similar for all four interpolators. This observation is confirmed in Figure 8.8. Figure 8.6 shows that the linear interpolation is systematically too low on convex terrain shapes and too high in concave terrain. Using the Coons interpolator this effect is less pronounced (Figure 8.7).

The three profiles reveal that linear interpolation tends to be above the surface on concave shapes and below the surface on concave shapes (Figures 8.10 - 8.12). The Coons interpolator, in contrast,

Table 8.11: Descriptive statistics for the deviations of the interpolated slopes from the slope of the first artificial surface.

	linear	Coons	Clough-Tocher	Smoothed Clough-Tocher
count	25921	25921	25921	25921
mean	3.43°	2.15°	2.17°	2.00°
minimum	0.00004°	0.00004°	0.00002°	0.00002°
maximum	45.33°	44.03°	44.03°	44.02°
standard deviation	5.25°	3.93°	3.94°	3.80°

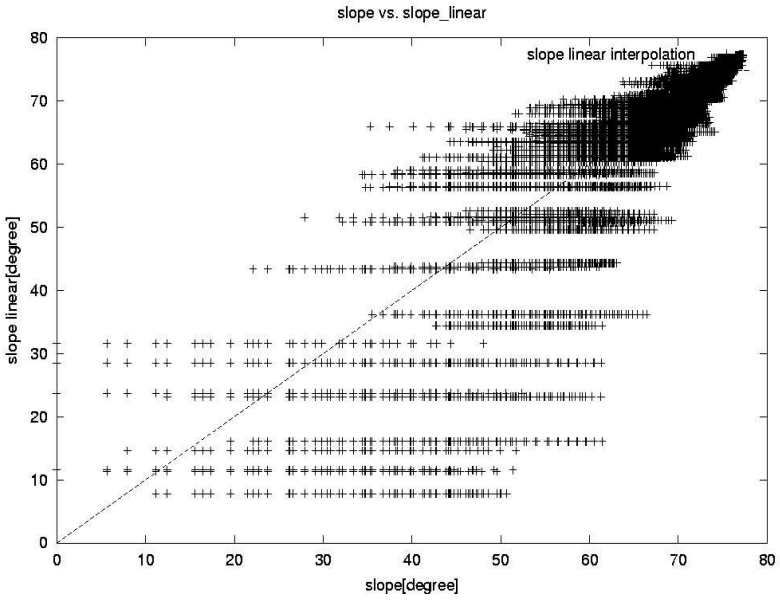

Figure 8.19: Slope of the linear interpolation compared to the slope of the first artificial function.

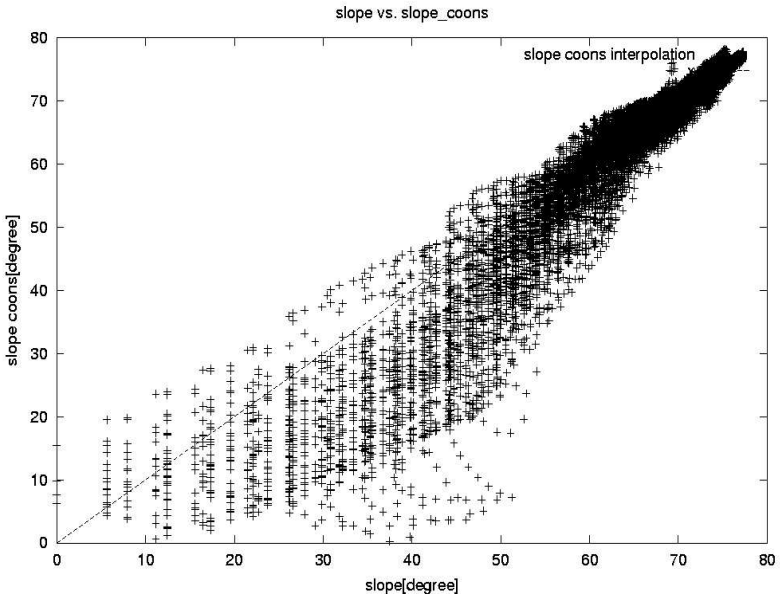

Figure 8.20: Slope of the Coons interpolation compared to the slope of the first artificial function.

changes more often from below to above the surface and the range of deviation is in general smaller. Therefore it is closer to the measured values at most profile sample points.

It can be noticed that the curves for linear and Coons interpolation have many peaks and spikes in common. This reflects that the same triangular tessellation has been used for both methods.

The comparison of the Coons interpolator with breakline extension with the interpolator without

Table 8.12: Statistics for the deviations of the Coons surfaces with and without breakline extension from the second artificial surface (only points with different elevations are considered).

	without breakline	with breakline
mean	3.91	1.88
standard deviation	6.34	6.20

Table 8.13: Comparison of the mean (unsigned) deviations of linear interpolation, Coons patch and bivariate quintic interpolation.

	linear	Coons	bivariate quintic
count	261	261	261
mean	36.5cm	28.3cm	28.6cm

extension reveals that the consideration of the breaklines constrains the interpolated surface closer to the field values. The better estimation of the surface normal not only affects the points on the road, but also those on either side of it.

Comparison with bivariate quintic surfaces

Bivaritate quintic interpolation has been introduced in Chapter 2. As this method is implemented in the ARC/INFO software, it is of practical relevance for many GIS users and a comparison with the methods developed in this thesis is therefore interesting. For this, the triangular tessellation has been created without considering the road casing as forced edges. This has been done because ARC/INFO uses another method for constrained lines than the software prototype of this thesis and thus using a normal Delaunay triangulation is the only method to ensure that the same tessellation is used for all the compared interpolators. Two interpolation points, both of which are very close to the convex hull have been removed because the bivariate quintic interpolation returned missing values. A perspective view of this bivariate quintic surface is shown in Figure 8.21. Table 8.13 shows the average deviations of linear interpolation, Coons patch and bivariate quintic interpolation.

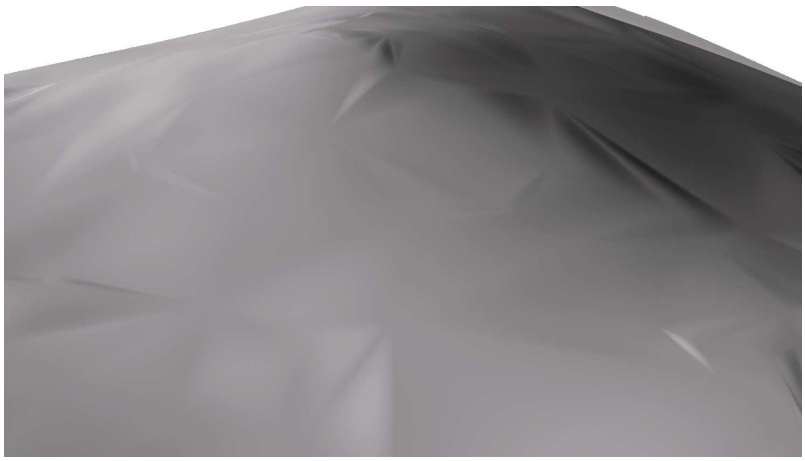

Figure 8.21: Test area interpolated with bivariate quintic interpolation over an unconstrained Delaunay triangulation (view from north-east).

The mean deviations for Coons patch and bivariate quintic interpolation are nearly the same. The stronger tendency of bivariate quintic interpolation to undulate, which was observed in Schneider (1998), does not appear in this test dataset. This is because the triangles are relatively well shaped and because the only sudden changes of slope in the test area are the road casings.

8.5 Discussion

The comparisons of the interpolators with artificial functions and with field measured data show that linear interpolation strongly tends to be below the surface in convex areas and above the surface in concave areas. They also reveal that this effect has a quantitatively important impact on the interpolation errors.

On the first artificial surface slopes have also been compared. Figures 8.19 and 8.20 show that the errors in slope decrease with increasing slope of the function. Since the function has more steep areas than flat ones, the mean deviations of slope are relatively small. Nevertheless, the errors of the Coons interpolator are in a smaller range for a given slope than those of the linear function.

All three cubic interpolators perform similarly in the quantitative comparisons, but the differences between the triangular Coons patch and the Clough-Tocher spline with linear cross-derivatives is extremely small. The smoothed version of the Clough-Tocher spline performs somewhat better on the artificial surface. In the field, such an effect could also be observed, but it is so small that it is lower than the round-off precision. In contrast to this, in the perspective views, the smoothed Clough-Tocher interpolator appears visually more pleasing than the other cubic methods, smoothing some artefacts of the linearly interpolated cross-derivatives.

The breakline extension was shown to be useful in this evaluation. It improves the quality of the surfaces considerably while the additional computation cost is low.

Comparing the results of the visual inspection with those of the comparison with artificial surfaces and the quantitative comparison with field data, two interesting points can be noticed:

- The cubic interpolators have a more realistic appearance in the 3D pictures than the linear interpolation and are in general considerably closer to the artificial function and the field measured values.
- The smoothed Clough-Tocher interpolation has a more realistic appearance in the 3D pictures than the Coons patch and the Clough-Tocher interpolation with linear cross-derivatives. However, in comparison with the artificial surface and the field measured data, there is little difference between the three cubic interpolators.

8.6 Conclusions

In this case study, the choice of a cubic interpolation method instead of the linear one reduces the mean deviation by approx. 25%. The main contribution to this difference stems from the sample points on the hilly landforms. Linear interpolation tends to underestimate the measured values in convex areas and thus performs worse than the cubic interpolators there. Such an effect would also occur on large concave landforms (the values of linear interpolation would then be significantly above the field measured value), but the convex forms are more marked in the test area than the concave ones. Surprisingly, on the road and on the plain, linear interpolation does not perform significantly better than the cubic methods (on the plain, it even performs slightly worse). This suggests that cubic

interpolators do not necessarily swing out on flat shapes if the surface normals at the triangle vertices are well estimated.

Comparison with the first artificial surface shows similar patterns, yet even more pronounced. One reason for this is the shape of the artificial surface which is relatively steep and has no flat areas where the linear interpolator might be expected to perform similar to the cubic interpolators.

The comparison of slopes on the first artificial surface shows that the cubic interpolators also yield better slope predictions. Scatter plots reveal that the uncertainties in slope (with all four methods) decrease with higher slope.

The quantitative analysis showed that there is only a limited difference between the three cubic interpolators. In practice, this would mean that the Clough-Tocher spline with linear interpolated cross-derivatives would often be the preferred choice, because it is mathematically easier than the other two methods and computationally less demanding. The smoothed Clough-Tocher interpolator, on the other, hand would be well suited where a surface has to be visually pleasing in shaded 3D-projections.

The breakline extension of the cubic interpolator proved to be useful and led to considerable improvement in the results for the second artificial surface. The smoothing of the Clough-Tocher interpolator had an influence on the visual appearance of the surface, while the influence on the quantitative results was small.

Chapter 9

Discussion

In this chapter, the research questions formulated in 1.2 are revisited. Section 9.3 addresses the question how linear features can be included in triangular Coons patches and Clough-Tocher Bézier splines. Section 9.2 discusses which artifacts occur, how they can be removed and whether this removal provides 'better' surfaces or not. Section 9.1 addresses the question whether the methods developed are 'better' then the ones already known in terrain modeling. Section 9.4 compares Coons patches and Clough-Tocher splines and discusses when which method should be used.

9.1 Assessment of the proposed methods

In Section 1.2, the question was raised whether the triangle based free-form surfaces used in this thesis are better than the ones already known in the area of digital terrain modeling. The empirical evaluation in chapter 8 was restricted to the comparison between Coons patches, Clough-Tocher Bézier splines, linear interpolation and bivariate quintic interpolation. Therefore, most of the text in this section compares these methods.

In contrast to Clough-Tocher splines, Coons patches allow for G_1-continuous integration of TINs with grids. Coons patches also provide more flexibility in the choice of the boundary curves and the blending functions. However, during the work on this thesis, no application field for this plus of flexibility could be identified. In the evaluation, the Coons patch used showed little difference to the Clough-Tocher Bézier splines. Additionally, it is more complex from a mathematical and an implementation point of view.

The evaluation in chapter 8 established that cubic G_1-continuous surfaces modelled the elevation of the study area and the artificial surface better than the linear surface because of estimating and considering the curvature of the terrain. An interesting question is whether the suitability of an interpolation method depends upon the terrain type, for instance whether G_1-continuous surfaces are better suited to model hilly landforms and linear facets are superior for modeling rugged mountainous terrain. Linear facets may imitate the visual appearance of a surface of the latter type well. However, even if a mountainous terrain is full of breaklines, the breaklines introduced by normal triangle edges are very likely to be at wrong locations from a geomophological point of view. Therefore, it is not a good idea to compensate for not sampled breaklines by using linear triangle facets. Instead, the breaklines relevant at the modeling scale should be captured. A G_1-continuous cubic free-form surface can be used to model these terrain breaks using the breakline extensions described in the chapters 4 and 5. These extensions are documented and allow for breakline modeling in cubic triangular surfaces instead of the black-box and closed source approach in quintic surfaces (ESRI, 2002).

The smoothing of the Clough-Tocher splines removed artifacts and improved visual appearance. However, in the evaluation, the influence of this extension on metric accuracy was small.

Compared with bivariate quintic interpolation the cubic interpolators showed approximately the same mean deviations in the comparison with field data. Bivariate quintic surfaces in general have a stronger tendency to undulate, as they are of degree five. As there are not many sudden changes in slope in the study area, this property did not have an influence on the results, however.

The refinement of the triangulation with the Ruppert algorithm has been assessed qualitatively. The method seems to be well suited to remove artifacts originating from small-angled triangles. As such artifacts are stronger the smoother an interpolator is constrained to be, the Ruppert algorithm is an interesting technique for smoothed Clough-Tocher Bézier splines or even more for G_2-continuous surfaces. The drawback of the algorithm is the additional amount of data which depends on the shape of the initial triangulation as well as on the specified minimum angle.

9.2 Removal of artifacts

Artifacts are landforms that are, based on knowledge about geomorphology, not plausible. If artifacts occur the available information including the geomorphological knowledge has not been sufficiently exploited. Research question 2 in section 1.2 deals with artifacts that may occur in triangle based Coons patches and Clough-Tocher Bézier splines. Research question 3 asks if the proposed artifact removal is successful. The following artifacts may occur in Coons patches and Clough-Tocher splines:

- Using linearly interpolated cross derivatives may generate prominent artifacts. For the Clough-Tocher Bézier splines this type of artifact can be reduced by using the smoothed version described in section 5.5.

- Long and thin triangles strongly accentuate undulation of cubic surfaces (Figure 6.1). Such triangles are common, even if the Delaunay criterion is used for triangulating. If the triangulation needs to be constrained (e. g. when triangulating contour data), such triangles are even more frequent. A possibility to avoid these artifacts is to make the triangle shapes more balanced by inserting new points, so called 'Steiner points' (Ruppert, 1993). The difficulty with this approach is that the data volume is enlarged. Furthermore, the interpolation of elevation values for these points may be of low quality if they are located in long and thin triangles themselves.

- The surfaces within the triangles are G_3-continuous using Coons patches and G_2-continuous using Clough-Tocher Bézier splines. Across the border between the triangles, the surface is only G_1-continuous in both cases. As a consequence, the curvature change is concentrated at the triangle edges, resulting in relatively stiff surfaces. In chapter 8, it has been shown that the elevation values of the cubic surfaces used in this thesis still tend to be below the terrain surface in convex areas (even if this effect is much weaker than for linearly interpolated surfaces). This is considered to be an effect of the stiff surfaces. A possibility to avoid such effects would be to use global G_2-continuous surfaces with Coons patches and Clough-Tocher elements of quintic degree. However, this approach would most probably increase the undulations in the case of long and thin triangles.

- Sinks in terrain are local minima of the surface. Sinks in digital terrain models are often artifacts. Hydrological applications are affected by such artificial sinks and thus plenty of literature exists describing how to cope with this problem (e. g. O'Callaghan and Mark (1984); Martinoni (1997)). Artificial sinks may be present due to several reasons:

- In flat areas, imprecise measurements may cause spurious sinks
- Artifacts of the data model may introduce artificial sinks. For instance, when using gridded data, sinks are usually present because the regular sampling scheme is not able to represent valley lines comprehensively.
- Sinks may occur as artifacts of the interpolation method.

Cubic surfaces may introduce spurious sinks of the latter type (Schneider, 1998), so linearly interpolated surfaces should be used for hydrologic applications. The removal of these artifacts would be quite complex. In the literature about shape modeling, the term 'monotonicity-preserving surface' exists for surface patches that do not have sinks if all the boundary curves are monotonously increasing or decreasing. However, in terrain modeling, boundary curves may not show monotonous behaviour, except if they belong to a drainage network. Maslva and Salkauskas (2000) show a method to create a boundary network from a hydrologic network and to force the boundary curves representing a drainage line to monotonicity. Unfortunately, monotonicity-preserving patch interpolation methods currently only exist for patches with four boundary curves (Costantini and Manni, 1996a,b).

Removal of artifacts does not necessarily mean that the resulting surface is more accurate. In this context, chapter 8 revealed interesting results. In the comparison with field data as well as in the comparison with the artificial surface, the G_1-continuous surfaces, on average, showed elevation values considerably closer to the reference surfaces than the linear interpolation. Thus, the removal of the planar facet artifact made the surface better suited for applications using elevations of a surface. In contrast, the smoothed Clough-Tocher Bézier splines did not show significant improvement compared to the methods using linear interpolated cross-derivatives. Even if the perspective views of the surfaces appeared more visually pleasing, the effect on the deviations was small.

9.3 Inclusion of linear information in Coons patches and Clough-Tocher Bézier splines

In chapter 1, it has been explained that the reconstruction of continuous surfaces from discrete input data is underdefined and thus introduces uncertainty. Therefore, it is important to use all available information to reduce the so-called shape uncertainty (Schneider, 2001a). If line data is available for terrain modeling, it is thus important to not only include the elevation of the line vertices into the specification of a continuous surface. The line information and the semantics associated with it are also information which can be used. Because of this, the research question 1 in Section 1.2 was how linear information be included in Coons patches and Clough-Tocher Bézier splines.

The most common forms of linear input data for terrain models are structure lines, contours and breaklines.

Structure lines are used to denote linear features on the surface which do not show an abrupt change in slope across them. Usually, geomorphological features like valleys and obtuse ridges are modelled as structure lines. Structure lines can be considered in Coons patches and Clough-Tocher Bézier splines by using a constrained Delaunay triangulation such that the structure line segments are not intersected by triangle edges.

Contours can be considered to be a special case of structure lines as they represent lines on the surface without a sudden change in slope across them. However, because of the implicit information contained in collections of isolines, special handling of contours is required. Because of this implicit

information, it is straightforward for a human to recognise terrain features by looking at a contour map. But it is very difficult to write computer programs which do the same. Contours have not been addressed in this thesis, because there exist a large number of publications covering this topic (Brändli, 1991; Heitzinger and Kager, 1998; Schneider, 1998; Thibault and Gold, 2000; Wise, 1997).

Breaklines describe linear features with a sudden change in slope across them. For instance sharp mountain ridges or borders of roads can be modelled as breaklines. The consideration of breaklines in otherwise G_1-continuous surfaces is less trivial than it initially may seem to be. In chapter 4, it has been shown that it is possible to abandon G_1-continuity for Coons patches only at breakline edges, without changing the boundary curve network. However, in doing so, the surface is still G_1-continuous at the breakline vertices and thus does not represent ridges well. Thus, the boundary curves on both sides of the breakline need to be changed to represent the surface break in a geomorphologically plausible way. Doing so, unintentional breaklines appear across these boundary curves, as the shape of the breakline has to be preserved. For Clough-Tocher Bézier splines, the situation is similar. Because of the subdivision, it can be chosen if the unintentional breaklines should occur within the macrotriangles or at the edges of macrotriangles.

How severe are the effects of this compromise for applications of terrain models? The sharpness of an unintentional breakline depends on the angle of the two breakline segments joining at a breakline vertex. It the angle is 180°, there is no unintentional breakline at all. The greater the deviation from 180°is, the sharper the unintentional breakline is. Note that the sharpness of unintentional breakline decreases with growing distance from the breakline vertex and finally disappears at the end of the boundary curve segment. Because of this, the influence of unintentional breaklines on the surface shape is minor (Figure 4.8).

Many applications benefit from the modeling of convexity and concavity provided by G_1-continuous surfaces. However, most of them can handle some unintentional breaklines. Applications requiring G_1-continuity need, for instance, to derive slope and aspect at every point, including the vertices and triangle edges. On intended and unintentional breaklines, slope and aspect are not defined. On intended breaklines, the derivation of slope and aspect makes no sense from a geomorphological point of view, so ambiguity is not a problem. On unintentional breaklines, in contrast, these derivatives may be of interest for an application. A workaround in this situation is to average the slope or aspect values of both sides. Because unintentional breaklines usually join in obtuse angles, this workaround provides a good approximation.

Which methods should be used for a proper solution not producing any unintentional breaklines? Coons patches or triangular splines with a degree higher than cubic, specified over the same tessellation do not solve the problem. To really solve it, the breakline should have only one tangent at the breakline vertices, which in general is not possible when using a triangular tessellation with straight line edges. Instead, a tessellation also capable of inserting cubic curves as edges would be necessary. Two problems would thus have to be solved:

1. The use of patches with boundary curves which are not necessarily straight lines in the orthogonal projection causes additional difficulties. The calculation of the parameter values for a point with given x and y coordinates is difficult and would require triangular Bézier clipping (Roth et al., 2000) for Clough-Tocher Bézier splines and isoparametric transformation to the standard triangle for Coons patches.

2. The creation of a tessellation ensuring that no straight edge intersects a curved edge is difficult. Boivin and Ollivier-Gooch (2002) recently presented a method to create tessellations with curved domain boundaries. To adapt this to the breakline problem, the triangulation do-

main would have to be split into several domains to have the breaklines at the boundaries. The rejoining of these disjoint parts then is a problem which remains to solve.

Thus, the compromise presented here is considered to be suitable for nearly all terrain applications. The development of a proper method is an interesting and ambitious task, but the benefit for terrain applications is considered to be small.

9.4 Comparison between Coons patches and Clough-Tocher Bézier splines

While the Coons patch method uses one surface for each triangle, the Clough-Tocher Bézier method splits triangles into three subtriangles. An important difference between the two methods is the cross-derivative function along the edges of the original triangles. In case of the cubic Coons patch, this function may be of any degree, while for the cubic Clough-Tocher Bézier spline, the cross-derivative function is at most quadratic. This difference is important if a surface consisting of rectangular cubic patches has to be joined in a G_1-continuous way with a surface consisting of cubic triangular patches. Because rectangular cubic patches, including cubic rectangular Bézier splines, have cubically interpolated cross-derivatives, this cannot be achieved with cubic Clough-Tocher Bézier triangles.

The boundary curves of cubic Clough-Tocher splines have to be Bézier curves. Coons patches provide more flexibility in this respect, as any form of curve can be used. Despite this flexibility, care has to be taken that the blending functions are compatible with the boundary curves.

Because of the Bernstein form, the triangular Clough-Tocher spline is much easier to implement than the triangular Coons patch. Once the elevations of the control points are determined, the formulae given in chapter 5 can be applied. In contrast, an implementation of the triangular Coons patch requires a lot of work to properly specify all the profiles of the ruled surfaces and especially to derive the equations for their derivatives.

The computational efficiency of the two methods is difficult to compare. The methods involve building a triangular tessellation and finding the triangle that the interpolation point is located in. These operations are the same for both methods. The creation time of a Delaunay triangulation depends to some extend on the algorithm used and can be accomplished in $O(n \log n)$ time, where n is the number of data points (Fortune, 1995). The search for a triangle in which a point is located can be done in $O(\log n)$ time by using the planar point location method (van Kreveld, 1997) or by using a heuristic. The computing time for the interpolation of the triangle patch itself, which is different for Coons patches and Clough-Tocher Bézier splines, is not dependent on the number of data points (O(1)). However, the computing cost of O may be different for the two methods. In the implementation made for this thesis, the Clough-Tocher method is about four times faster than the Coons patch. However, this depends enormously on how the implementation is made and is only partly dependent on the interpolation methods.

The evaluation in this thesis established, that there are only minor difference of surface shape between Coon patch and Clough-Tocher splines. Therefore, the Clough-Tocher splines are the better choice for most programmers, as the implementation is easier. The Coons patch in contrast is better suited if rectangular patches are present in the same surface. It also provides more flexibility, but in this thesis, no need to make use of this flexibility has been found. However, very special tasks may benefit from it, for instance the avoidance of spurious sinks.

Chapter 10

Conclusions

The central goal of this thesis was to adapt triangular Coons patches and triangular Clough-Tocher Bézier splines for use in terrain modelling. In turn, this required linear information to be included by using constrained Delaunay triangulations and by developing methods to consider breaklines in the surfaces. Several kinds of artifacts have been identified and two methods to soften these artifacts have been applied to terrain modelling: the smoothed version of the Clough-Tocher Bézier spline to remove the artifacts arising from linearly interpolated cross-derivatives and the Ruppert algorithm for mesh refinement to avoid artifacts because of long and thin triangles. This chapter summarises the main achievements and insights of this work and closes with an outlook to possible further research.

10.1 Achievements

- Triangular and cubic Coons patches have been applied. Each triangle surface patch has been modelled with three cubic boundary curves and a patch with linearly interpolated cross-derivatives. The Coons patch technique has not been available for digital terrain modelling until now.
- Breakline extensions have been developed for triangular Coons patches and cubic Clough-Tocher Bézier splines. These extensions change the boundary curves leading to the inserted breakline to force a surface break. They maintain the shape of the boundary curves representing the breakline and ensure that there is no gap in the surface (G_0-continuity).
- To soften artifacts arising from linearly interpolated cross-derivatives, the technique of smoothing Clough-Tocher Bézier splines, developed in the discipline of computer aided geometric design (CAGD), has been implemented and applied to terrain modelling.
- To avoid long and thin triangles in the triangulation, the Ruppert algorithm, which inserts additional points into the triangulation, has been implemented. Possible problems of this algorithm for terrain modeling have been discussed and perspective views of digital terrain surfaces have been used to visualise its effects.
- The triangular interpolators used in this thesis and linear interpolation have been evaluated. For this, interpolated elevation values have been compared with field measured data and with artificial surfaces.
- All the methods which have been applied and developed in this thesis have been implemented in a prototype software using C++.

10.2 Insights

Assessment of the developed methods The triangular cubic Coons patches and the cubic Clough-Tocher Bézier splines used and extended in this thesis have been compared with linear interpolation and bivariate quintic surfaces on artificial surfaces and on a real terrain. The nonlinear interpolators predicted the shape considerably better on both the artificial surface and the study area. The differences between the three cubic interpolators used, Coons patches, Clough-Tocher Bézier splines with linearly interpolated cross-derivatives and smoothed Clough-Tocher splines were found to be small. The quintic interpolation shows differences to the cubic interpolators, but in terms of quality, all the higher order methods performed similarly. The breakline extensions on the cubic interpolators clearly lead to an improvement of the predicted results.

Artifacts Several kinds of artifacts may appear when using triangle based free form surfaces. Long and thin triangles strongly accentuate undulations in surfaces. By using the Ruppert algorithm, these triangles can be removed by inserting additional points into the triangulation. The disadvantage of this approach, however, is that it increases the data volume. Using Coons patches and Clough-Tocher Bézier splines with linearly interpolated cross-derivatives generates artifacts at the border of the triangles. These artifacts can be avoided by using smoothed Clough-Tocher splines, although the evaluation suggested that, for the case study used, the visual impact of the smoothing may be more important than the numerical effects.

Linear information Because of the triangle based approach, the cubic Coons patches and Clough-Tocher Bézier splines are generally well suited to incorporate linear information. One possible technique is to use constrained Delaunay triangulations. Sharp surface breaks can also be modelled by altering the surface normals at the vertices of the triangle patches adjacent to the breaklines. However, if the surface should not have gaps across the breaklines (G_0-continuity), the choice of the normals needs to be carried out carefully. In this case, the price of the insertion of a breakline may be the appearance of unintentional breaklines at some triangle edges leading to the intentional breakline.

Coons patches compared to Clough-Tocher Bézier splines Cubic Coons patches and cubic Clough-Tocher Bézier splines are very similar if the interpolation function of the cross-derivatives is the same (e. g. linear interpolation). The most important differences between these two methods are that the cubic Coons patch can have a cross-derivative function of cubic degree whereas this function can be at most quadratic for cubic Clough-Tocher Bézier splines. Therefore Clough-Tocher splines cannot be applied if a tessellation consists of triangular and quadrilateral patches and if this surface has to be G_1-continuous. Also, the cubic Coons patch allows for any kind of boundary curves while Clough-Tocher Bézier splines are restricted to Bézier curves. The advantage of the Clough-Tocher splines is that they are much easier to handle from a mathematical and from an implementation point of view.

10.3 Outlook

10.3.1 Applying continuous surface models

A considerable amount of research on the methodological aspects of continuous terrain models has been accomplished to date. However, most digital terrain modelling tasks, besides terrain visualisation, are performed with discrete terrain models. Rare examples of work using continuous models are

Wood (1998) and Tucker et al. (2001). The latter is one of the very few examples using irregularly distributed data points. So, there is definitely a need for research comparing the properties of continuous and discrete models from the perspective of specific applications. Such research could reveal under which circumstances and for which applications it is worthwhile to use continuous models. This could greatly promote the use of continuous models in areas where such models are found to be suitable.

10.3.2 Direct derivation of complex nonlocal information

As mentioned in Chapter 1, an ideal situation for terrain modeling would be to specify one continuous surface model and to derive all required terrain information directly from this surface model. For Coons patches and Clough-Tocher Bézier splines, local terrain information based on elevation or derivatives can be calculated directly from the surface. Examples for such local terrain information are elevation, gradient, aspect or curvature. However, there are a lack of methods to derive more complex terrain derivatives from such surfaces, for instance the visible area (viewshed) of a point or the catchment draining into an area. Most of these complex terrain derivatives can be calculated directly from linearly interpolated surfaces. Thus, for the cubic surfaces described in this thesis, complex nonlocal information has to be derived by approximating the cubic surface, either with many linearly interpolated triangles or by transforming the cubic surface to a dense grid. Then, the extraction methods for the linear triangles or for grids can be used. However, with these approaches, memory requirements and computation time may be high if the resolution is very high. Furthermore, it is hard to determine an appropriate resolution.

Deriving complex nonlocal information directly from continuously differentiable surfaces is an interesting task for future research. Although it is quite complex, approaches already exist. The following two paragraphs mention existing approaches and possible research tasks for viewshed calculations and the derivation of hydrological information, two important examples of complex nonlocal information.

Viewshed Intervisibility between two points on Clough-Tocher splines can be calculated using triangular Bézier clipping (Roth et al., 2000). This technique uses an approximation approach to find the intersection point between a ray and a Bézier patch. Even if it is approximate, the amount of deviation to the accurate point can be specified. However, the task of finding the area visible from a given point is still unsolved.

Derivation of hydrological information An important terrain derivative for hydrological modeling is the path of steepest descent, also referred to as a streamline or fall line. On a linearly interpolated TIN, the path of steepest descent can be calculated directly because it does not change its direction within a triangle facet. On continuously differentiable surfaces, in contrast, a path of steepest descent changes its direction continuously. Approximation of the path of steepest descent with many straight line segments is possible but has disadvantages, for example waste of memory and zig-zag lines in valleys. Direct derivation of streamlines from vector fields can be calculated by solving a differential equation. It would be interesting to see whether this technique could also be applied to directly derive paths of steepest descent from cubic Coons patches and Clough-Tocher splines.

An interesting property of continuously differentiable surfaces is that two paths of steepest descent never join. The catchment area of a point therefore is a line and its area is 0. Nevertheless, the calculation of catchment areas for pits or contour line segments would be possible by using surface networks. A surface network is a graph containing critical points (passes, peaks, pits) and critical lines (valleys, ridges) of a surface (Pfaltz, 1976). Surface networks can be derived by identifying passes

and building the connections to the related peaks and pits (Martinoni, 1997; Wood, 1998). Thus, in order to derive surface networks from cubic Coons patches and Clough-Tocher Bézier splines, it is necessary to find passes, peaks, pits in such surfaces and to derive streamlines from them.

10.3.3 Tessellation with arbitrary curves

As already discussed in Section 9.3, a tessellation with the possibility to incorporate curved segments would give more possibilities in including terrain features. For instance, breaklines could be considered without unintentional breaklines (Chapter 4). Besides the construction of such a triangulation, the mapping from real world coordinates to parameter space, which is necessary prior to interpolation, is an additional problem.

10.3.4 Further Comparisons of interpolation methods with field data

Further comparisons of interpolated values with field data would be interesting. The evaluation in Chapter 8 could be extended by other experiments in different ways:

- Comparison with field values of other study areas may possibly lead to different results than the evaluation in Chapter 8. Therefore more such experiments are desirable.
- Other interpolators described in Chapter 2 could be used for comparison, for instance minimum curvature splines and finite elements. In this thesis, these methods have not been used for comparison because they are either not implemented in commonly used GIS packages or the implementations only support interpolation to a raster and not to irregularly spaced interpolation points.
- Most GIS users are interested in terrain derivatives, for instance slope, curvature or viewshed, rather than in the elevation values itself. Therefore it would also be interesting to compare such derivatives in a field study.

Bibliography

Akima, H. (1978). A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Transactions on Mathematical Software*, 4(2):148–159.

Barnhill, R. E. and Gregory, J. A. (1975). Compatible smooth interpolation in triangles. *Journal of Approximation Theory*, 15:214–225.

Beach, R. (1991). *Introduction to the Curves and Surfaces of Computer-Aided Design*. Van Nostrand Reinhold Computer Library, New York.

Bern, M. and Eppstein, D. (1995). Mesh Generation And Optimal Triangulation. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*. World Scientific, Singapore.

Boivin, C. and Ollivier-Gooch, C. (2002). Guaranteed-quality triangular mesh generation for domains with curved boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185 – 1213.

Bolstad, P. and Stowe, T. (1994). An evaluation of DEM accuracy: elevation, slope and aspect. *Photogrametric Engineering and Remote Sensing*, 60(11):1327–1332.

Brändli, M. (1991). Oberflächeninterpolation aus Höhenkurvendaten. Master's thesis, Department of Geography, University of Zurich.

Burrough, P. A. and McDonnell, R. A. (1998). *Principles of Geographic Information Systems*. Oxford University Press, Oxford.

Carter, J. (1992). The effect of data precision on the calculation of slope and aspect using gridded DEMs. *Cartographica*, 29(1):22–34.

Chiles, J.-P. and Delfiner, P. (1999). *Geostatistics*. Wiley, New York.

Corripio, J. (2003). Vectorial algebra algorithms for calculating terrain parameters from DEMs and solar radiation modelling in mountainous terrain. *International Journal of Geographic Information Science*, 17(1):1–23.

Costantini, P. and Manni, C. (1996a). A bicubic shape-preserving blending scheme. *Computer Aided Geometric Design*, 13:307–331.

Costantini, P. and Manni, C. (1996b). Monotonicity-preserving interpolation of nongridded data. *Computer Aided Geometric Design*, 13:467–495.

de Floriani, L. and Puppo, E. (1992). An On-Line Algorithm for Constrained Delaunay Triangulation. *Computer Visions, Graphics and Image Processing: Graphical Models and Image Processing*, 54(3):290–300.

Dirnböck, T., Dullinger, S., and Grabherr, G. (2003). A regional impact assessment of climate and land-use change on alpine vegetation. *Journal of Biogeography*, 30(3):401–417.

Dubrule, O. (1984). Comparing splines and kriging. *Computers&Geosciences*, 10:327–338.

Ebner, H. (1983). Berücksichtigung der lokalen Geländeform bei der Höheninterpolation mit finiten Elementen. *Bildmessung und Luftbildwesen*, 51(1):3–9.

Ebner, H., Hofmann, B., Reiss, P., and Steidler, F. (1980). HIFI - a minicomputer program package for height interpolation by finite elements. *International Archives of Photogrammetry and Remote Sensing*, 23:202–215.

Ebner, H. and Reiss, P. (1978). Height interpolation by the method of finite elements. In *Proceedings of the Digital Terrain Modelling Symposium, St. Louis*, pages 241–254.

Erxleben, J., Elder, K., and Davis, R. (2002). Comparison of spatial interpolation methods for estimating snow distribution in the colorado rocky mountains. *Hydrological Processes*, 16:3627–3649.

ESRI (2002). Arc/info help.

Farin, G. (1985). A modified Clough-Tocher interpolant. *Computer Aided Geometric Design*, 2(1-3):19–27.

Farin, G. (1997). *Curves and Surfaces for CAGD. A practical guide*. Academic press, San Diego, London, Boston, New York, Sydney, Tokyo, Toronto, fourth edition edition.

Farin, G., Hoschek, J., and Kim, M. (2002). *Handbook of Computer Aided Geometric Design*. North-Holland, Amsterdam.

Floriani, L. D. and Magillo, P. (1996). Representing the visibility structure of a terrain through a nested horizon map. *International Journal of Geographic Information Science*, 10(5):541–562.

Fortune, S. (1995). Voronoi diagrams and Delaunay triangulations. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 225–265. World Scientific, Singapore.

Fricker, H., Hyland, G., Coleman, R., and Young, N. (2000). Digital elevation models for the Lambert Glacier-Amery Ice Shelf system, East Antarctica, from ERS-1 satellite radar altimetry. *Journal of Glaciology*, 46(155):553–560.

Gallant, J. and Wilson, J. (2000). Primary topographic attributes. In Wilson, J. and Gallant, J., editors, *Terrain analysis*. John Wiley & sons, London.

Gao, J. (1997). Resolution and accuracy of terrain representation by grid DEMs at a micro-scale. *International Journal of Geographic Information Sciences*, 11(2):199–212.

Giles, P. and Franklin, S. (1996). Comparison of derivative topographic surfaces of a DEM generated from stereoscopic SPOT images with field measurements. *Photogrametric Engineering and Remote Sensing*, 62(10):1165–1171.

Goodchild, M. F. (1992). Geographical data modeling. *Computers & Geosciences*, 18(4):401–408.

GRASS (2004). Grass 5.3.x reference manual. http://grass.itc.it/gdp/html_grass5/index.html.

Heitzinger, D. and Kager, H. (1998). High quality DTMs from contourlines by knowledge-based classification of problem regions. http://www.ipf.tuwien.ac.at/publications/dh_p_isprs98/sh_p_isprs98.html.

Heller, M. (1990). Triangulation algorithms for adaptive terrain modeling. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, volume 1, pages 163–174, Zurich, Switzerland.

Herzfeld, U. (1999). Geostatistical interpolation and classification of remote sensing data from ice surfaces. *International Journal of Remote Sensing*, 20(2):307–327.

Horn, B. (1981). Hill shading and the reflectance map. *Proceedings of the IEEE*, 69(1):14–47.

Hugentobler, M. (2000). Fortpflanzung von unsicherheiten in dreiecksbasierten digitalen Geländemodellen. Master's thesis, Department of Geography, University of Zurich.

Hugentobler, M. (2001). Propagation of uncertainties in digital terrain models with interval methods. In *Proceedings of the GIS Research UK 9th Annual Conference*, pages 341–44, Glamorgan, Wales.

Hugentobler, M. (2002). Interpolation of continuous surfaces for terrain modeling with Coons patches. In *Proceedings of GISRUK 2002*, pages 13–15.

Hugentobler, M., Purves, R., and Schneider, B. (2004). Evaluating methods for interpolating continuous surfaces from irregular data: a case study. Accepted as conference paper for Spatial Data Handling 2004.

Hugentobler, M. and Schneider, B. (2004). Breaklines in coons surfaces over triangles for the use in terrain modelling. Submitted to Computers & Geosciences.

Hutchinson, M. and Gallant, J. (2000). Digital elevation models and representation of terrain shape. In Wilson, J. and Gallant, J., editors, *Terrain analysis*. John Wiley & sons, London.

Isaaks, E. and Srivastava, M. (1989). *An introduction to applied geostatistics*. Oxford University Press, Oxford.

Jimenez-Espinosa, R. and Chica-Olmo, M. (1999). Application of geostatistics to identify gold-rich areas in the finisterre-fervenza region, nw Spain. *Applied Geochemistry*, 14:133–145.

Kashyap, P. (1996). Improving Clough-Tocher interpolants. *Computer Aided Geometric Design*, 13(7):629–651.

Klucewicz, I. M. (1978). A piecewise c1-interpolant to arbitratily spaced data. *Computer Graphics and Image Processing*, 8:92–112.

Martinez-Cob, A. (1996). Multivariate geostatistical analyses of evapotranspiration and precipitation in mountanious terrain. *Journal of Hydrology*, 174:19–35.

Martinoni, D. (1997). Extraktion von hydrologischen Strukturen aus triangulierten Geländemodellen. Master's thesis, Department of Geography, University of Zurich.

Martinoni, D. (2002). *Models and experiments for quality handling in digital terrain modelling*. PhD thesis, Department of Geography, University of Zurich.

Maslva, L. and Salkauskas, K. (2000). Enforced drainage terrain models using minimum norm networks and smoothing splines. *Rocky Mountain Journal of Mathematics*, 30(3):1075–1109.

Matheron, G. (1962). *Le krigeage I*. Number 14 in Mémoires du bureau de recherches géologiques et minières. Editions Technip, Paris.

Matheron, G. (1963). *Le krigeage II*. Number 24 in Mémoires du bureau de recherches géologiques et minières. Editions Technip, Paris.

Miller, G., Pav, S., and Walkington, N. (2003). When and why Ruppert's algorithm works. In *Proceedings of the 12th International Meshing Roundtable*, pages 91–102.

Mitas, L. and Mitasova, H. (1999). Spatial interpolation. In P.Longley, Goodchild, M., Maguire, D., and D.W.Rhind, editors, *Geographical Information Systems*, pages 481–492. Longman, London.

O'Callaghan, J. and Mark, D. (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, 28:323–344.

Pac, R. (2000). X-sar/srtm shuttle radar topography mission. http://www2.dlr.de/oeffentlichkeit/specials/sonderseiten/srtm/srtm_folder_02.pdf.

Pebesma, E. (2000). gstat user's manual. http://www.gstat.org/bin/gstat-2.1.0.a4.ps.gz.

Peucker, T., Fowler, R., Little, J., and Mark, D. (1978). The triangulated irregular network. In *Proceedings of the Digital Terrain Model Symposium*, pages 516–540, St. Louis, Missouri.

Pfaltz, J. (1976). Surface networks. *Geographical Analysis*, 8(1):77–93.

Pfeifer, N. (2002). *3D terrain models on the basis of a triangulation*. PhD thesis, Technische Universität Wien.

Pfeifer, N., Stadler, P., and Briese, C. (2001). Derivation of digital terrain models in the SCOP++ environment. In *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Terrain Models*, Stockholm.

Piegl, L. and Tiller, W. (1997). *The NURBS book*. Springer, Berlin.

Preusser, A. (1984). Bivariate Interpolation über Dreieckselementen durch Polynome 5. Ordnung mit C1-Kontinuität. *Zeitschrift fuer Vermessungswesen*, 6:292–301.

Rogers, D. and Adams, J. (1990). *Mathematical elements for computer graphics*. McGraw-Hill, London.

Roth, M., Diezi, P., and Gross, M. (2000). Triangular Bezier clipping. technical report 347, ETH Zürich.

Ruppert, J. (1993). A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92.

Schneider, B. (1998). *Geomorphologisch plausible Rekonstruktion der digitalen Repräsentation von Geländeoberflächen aus Höhenliniendaten*. PhD thesis, Department of Geography, University of Zurich.

Schneider, B. (2001a). On the uncertainty of local shape of lines and surfaces. *Cartography and Geographic Information Science*, 28(4):237–247.

Schneider, B. (2001b). Phenomenon-based specification of the digital representation of terrain surfaces. *Transactions in GIS*, 5(1):39–52.

Skidmore, A. (1989). A comparison of techniques for calculating gradient and aspect from a gridded digital elevation model. *International Journal of Geographical Information Systems*, 3(4):323–334.

Song, W. and Haithcoat, T. (2003). Improve the positional accuracy of digital parcel map through vector migration. http://www.urisa.org/Journal/Under_Review/song/improve_the_positional_accuracy.htm.

Thibault, D. and Gold, C. (2000). Terrain reconstruction from contours by skeleton construction. *GeoInformatica*, pages 349–373.

Tobler, W. (1970). A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46(2):234–240.

Tucker, G., Lancaster, S., Gasparini, N., Bras, R., and Rybarczyk, S. (2001). An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks. *Computers & Geosciences*, 27(8):959–973.

Unbenannt, M. (1999). Generation and Analysis of High-Resolution Digital Elevation Models for Morphometric Relief Classification, represented at a Cuesta Scarp Slope on the Colorado Plateau, USA. In *Proceedings of the 3rd German-Dutch Symposium KvAG (Niederlande) / DGPF-AK "Interpretation von Fernerkundungsdaten"*.

van Kreveld, M. (1997). Algorithms for triangulated terrains. In *Proceedings of the 24th SOFSEM*, Lecture Notes in Computer Sciences, pages 19–36, Berlin. Springer Verlag.

Wackernagel, H. (1998). *Multivariate geostatistics*. Springer, New York.

Watson, D. (1992). *Contouring*. Pergamon Press, Oxford.

Weibel, R. and Brändli, M. (1995). Adaptive methods for the refinement of digital terrain models for geomorphometric applications. *Zeitschrift für Geomorphologie*, Supplementband 101:13–30.

Weibel, R. and Heller, M. (1990). Digital terrain modelling. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographical information systems - principles and applications*, pages 269–297. Longman, London.

Wise, S. (1997). The effect of gis interpolation errors on the use of digital elevation models in geomorphology. In Lane, S., Richards, K., and Chandler, J., editors, *Landform monitoring, modelling and analysis*, pages 139–164. John Wiley & Sons Ltd., London.

Woo, M., Neider, J., Davis, T., and Shreiner, D. (1999). *OpenGL programming guide*. Addison-Wesley, Reading, Massachusetts.

Wood, J. (1998). Modelling the continuity of surface form using digital elevation models. In *Proceedings of the 8th International Symposium on Spatial Data Handling*, pages 725–736.

Wood, J. and Fisher, P. (1993). Assessing interpolation accuracy in elevation models. *IEEE Computer Graphics & Applications*, pages 48–56.

Zevenbergen, L. and Thorne, C. (1987). Quantitative analysis of land surface topography. *Earth Surface Processes and Landforms*, 12:47–56.

Appendix A

Installation of the software protorype 'Tritemo'

The source code of 'Tritemo' can be downloaded from *http://www.geo.unizh.ch/gis/research/dtm/tritemo.tar.gz*. *tar -xzvf* unpacks the code. Note that for compilation, the qt library (version3) needs to be installed (including the header files) as well as the OpenGL or Mesa library and headers. Then, for compilation, two environment variables need to be set: BREAKDIR=path to tritemo directory and QTDIR=path to qt directory (usually /usr/lib/qt3). Then change to the tritemo directory and run make. Note that 'Tritemo' is a prototype and it may be necessary to make changes in the makefile manually if there are errors during the compilation. If the compilation was sucessfull, there is an executable 'tritemo' in the tritemo directory.

In the user interface of 'Tritemo', triangulations can be created by opening dxf-files and triangulations can be stored and read in taf-files. Note: the code for dxf reading was developed with the purpose of reading the author's test files. If you have any problems with the files you are using, it may be necessary to derive a new class from the FileReader interface and implementing FileReader::readFile yourself (or to change DxfReader::readFile). Once a triangulation has been created, it can be manipulated in the user interface by swapping edges or entering additional points. The z-coordinate of these additional points are the ones of the actual surface. The shape of the surface can be visualised by entering the coordinates of the view position and the coordinates of the point the center of the perspective picture. Appendix B contains descriptions of the classes related to triangulation and interpolation (for those related to GUI, file handling, 2D- and 3D-visualisation see the documentations in the source code generated by Doxygen).

Appendix B

Code documentation

B.1 Triangulation Class Reference

B.1.1 Description

Interface for Triangulation classes

B.1.2 Member Enumeration Documentation

enum Triangulation::forcedCrossBehaviour

Enumeration describing the behaviour, if two forced lines cross. SNAP_TO_VERTICE means, that the second inserted forced line is snapped to the closest vertice of the first inserted forced line. DELETE_FIRST means, that the status of the first inserted forced line is reset to that of a normal edge (so that the second inserted forced line remain and the first not

B.1.3 Member Function Documentation

virtual void Triangulation::addLine (Line3D * *line*, bool *breakline*) `[pure virtual]`

Adds a line (e.g. a break-, structure- or an isoline) to the triangulation

virtual int Triangulation::addPoint (Point3D * *p*) `[pure virtual]`

Adds a point to the triangulation

virtual bool Triangulation::calcNormal (double *x*, double *y*, Vector3D * *result*) `[pure virtual]`

Calculates the normal at a point on the surface and assigns it to 'result'. Returns true in case of success and flase in case of failure

virtual bool Triangulation::calcPoint (double *x*, double *y*, Point3D * *result*) `[pure virtual]`

Calculates x-, y and z-value of the point on the surface and assigns it to 'result'. Returns true in case of success and flase in case of failure

virtual void Triangulation::draw (QPainter * *p*, double *xlowleft*, double *ylowleft*, double *xupright*, double *yupright*, double *width*, double *height*) const [pure virtual]

draws the points, edges and the forced lines

virtual void Triangulation::eliminateHorizontalTriangles () [pure virtual]

Eliminates the horizontal triangles by swapping

virtual int Triangulation::getNumberOfPoints () [pure virtual]

Returns the number of points

virtual int Triangulation::getOppositePoint (int *p1*, int *p2*) [pure virtual]

Returns the number of the point opposite to the triangle points p1, p2 (which have to be on a halfedge)

virtual Point3D* Triangulation::getPoint (unsigned int *i*) const [pure virtual]

Returns a pointer to the point with number i. Any virtual points must have the number -1

virtual QValueList<int>* Triangulation::getPointsAroundEdge (double *x*, double *y*) [pure virtual]

Returns a value list with the numbers of the four points, which would be affected by an edge swap. This function is e.g. needed by **NormVecDecorator** (p. **??**) to know the points, for which the normals have to be recalculated. The list has to be deleted by the code which calls this method

virtual QValueList<int>* Triangulation::getSurroundingTriangles (int *pointno*) [pure virtual]

Returns a pointer to a value list with the information of the triangles surrounding (counterclockwise) a point. Four integer values describe a triangle, the first three are the number of the half edges of the triangle and the fourth is -10, if the third (and most counterclockwise) edge is a breakline, and -20 otherwise. The value list has to be deleted by the code which called the method. Any virtual point needs to have the number -1

virtual bool Triangulation::getTriangle (double *x*, double *y*, Point3D * *p1*, Point3D * *p2*, Point3D * *p3*) [pure virtual]

Finds out, in which triangle the point with coordinates x and y is and assigns the points at the vertices to 'p1', 'p2' and 'p3

virtual bool Triangulation::getTriangle (double *x*, double *y*, Point3D * *p1*, int * *n1*, Point3D * *p2*, int * *n2*, Point3D * *p3*, int * *n3*) [pure virtual]

Finds out, in which triangle the point with coordinates x and y is and assigns the numbers of the vertices to 'n1', 'n2' and 'n3' and the vertices to 'p1', 'p2' and 'p3'

virtual double Triangulation::getXMax () [pure virtual]

Returns the largest x-coordinate value of the bounding box

virtual double Triangulation::getXMin () [pure virtual]

Returns the smallest x-coordinate value of the bounding box

virtual double Triangulation::getYMax () [pure virtual]

Returns the largest y-coordinate value of the bounding box

virtual double Triangulation::getYMin () [pure virtual]

Returns the smallest x-coordinate value of the bounding box

virtual void Triangulation::performConsistencyTest () [pure virtual]

Performs a consistency check, remove this later

virtual bool Triangulation::pointInside (double *x*, double *y*) [pure virtual]

Returns true, if the point with coordinates x and y is inside the convex hull and false otherwise

virtual bool Triangulation::readFromTAFF (QString *filename*) [pure virtual]

Reads the content of a taff-file

virtual void Triangulation::ruppertRefinement () [pure virtual]

Adds points to make the triangles better shaped (algorithm of ruppert)

virtual bool Triangulation::saveToTAFF (QString *filename*) const [pure virtual]

Saves the content to a taff file

virtual void Triangulation::setBreakEdgeColor (int *r*, int *g*, int *b*) [pure virtual]

Sets the color of the breaklines

virtual void Triangulation::setEdgeColor (int *r*, int *g*, int *b*) [pure virtual]

Sets the color of the normal edges

virtual void Triangulation::setForcedCrossBehaviour (Triangulation::forcedCrossBehaviour *b*) [pure virtual]

Sets the behaviour of the triangulation in case of crossing forced lines

virtual void Triangulation::setForcedEdgeColor (int *r*, int *g*, int *b*) `[pure virtual]`

Sets the color of the forced edges

virtual void Triangulation::setTriangleInterpolator (TriangleInterpolator * *interpolator*) `[pure virtual]`

Sets an interpolator object

virtual bool Triangulation::swapEdge (double *x*, double *y*) `[pure virtual]`

Swaps the edge which is closest to the point with x and y coordinates (if this is possible)

B.2 TriDecorator Class Reference

B.2.1 Description

Decorator class for Triangulations (s. Decorator pattern in Gamma et al.)

B.2.2 Member Function Documentation

virtual void TriDecorator::addLine (Line3D * *line*, bool *breakline*) `[virtual]`

Adds a line (e.g. a break-, structure- or an isoline) to the triangulation

virtual int TriDecorator::addPoint (Point3D * *p*) `[virtual]`

Adds a point to the triangulation

void TriDecorator::addTriangulation (Triangulation * *t*) `[inline, virtual]`

Adds an association to a triangulation

virtual bool TriDecorator::calcNormal (double *x*, double *y*, Vector3D * *result*) `[virtual]`

Calculates the normal at a point on the surface and assigns it to 'result'. Returns true in case of success and flase in case of failure

virtual bool TriDecorator::calcPoint (double *x*, double *y*, Point3D * *result*) `[virtual]`

Calculates x-, y and z-value of the point on the surface and assigns it to 'result'. Returns true in case of success and flase in case of failure

virtual void TriDecorator::draw (QPainter * *p*, double *xlowleft*, double *ylowleft*, double *xupright*, double *yupright*, double *width*, double *height*) const `[virtual]`

draws the points, edges and the forced lines

virtual void TriDecorator::eliminateHorizontalTriangles () [virtual]

Eliminates the horizontal triangles by swapping

virtual int TriDecorator::getNumberOfPoints () [virtual]

Returns the number of points

virtual int TriDecorator::getOppositePoint (int *p1*, int *p2*) [virtual]

Returns the number of the point opposite to the triangle points p1, p2 (which have to be on a halfedge)

virtual Point3D* TriDecorator::getPoint (unsigned int *i*) const [virtual]

Returns a pointer to the point with number i. Any virtual points must have the number -1

virtual QValueList<int>* TriDecorator::getPointsAroundEdge (double *x*, double *y*) [virtual]

Returns a value list with the numbers of the four points, which would be affected by an edge swap. This function is e.g. needed by **NormVecDecorator** (p. **??**) to know the points, for which the normals have to be recalculated. The list has to be deleted by the code which calls this method

virtual QValueList<int>* TriDecorator::getSurroundingTriangles (int *pointno*) [virtual]

Returns a pointer to a value list with the information of the triangles surrounding (counterclockwise) a point. Four integer values describe a triangle, the first three are the number of the half edges of the triangle and the fourth is -10, if the third (and most counterclockwise) edge is a breakline, and -20 otherwise. The value list has to be deleted by the code which called the method. Any virtual point needs to have the number -1

virtual bool TriDecorator::getTriangle (double *x*, double *y*, Point3D * *p1*, Point3D * *p2*, Point3D * *p3*) [virtual]

Finds out, in which triangle the point with coordinates x and y is and assigns the points at the vertices to 'p1', 'p2' and 'p3

virtual bool TriDecorator::getTriangle (double *x*, double *y*, Point3D * *p1*, int * *n1*, Point3D * *p2*, int * *n2*, Point3D * *p3*, int * *n3*) [virtual]

Finds out, in which triangle the point with coordinates x and y is and assigns the numbers of the vertices to 'n1', 'n2' and 'n3' and the vertices to 'p1', 'p2' and 'p3'

virtual double TriDecorator::getXMax () [virtual]

Returns the largest x-coordinate value of the bounding box

virtual double TriDecorator::getXMin () [virtual]

Returns the smallest x-coordinate value of the bounding box

virtual double TriDecorator::getYMax () [virtual]

Returns the largest y-coordinate value of the bounding box

virtual double TriDecorator::getYMin () [virtual]

Returns the smallest x-coordinate value of the bounding box

virtual void TriDecorator::performConsistencyTest () [virtual]

Performs a consistency check, remove this later

virtual bool TriDecorator::pointInside (double *x*, double *y*) [virtual]

Returns true, if the point with coordinates x and y is inside the convex hull and false otherwise

virtual bool TriDecorator::readFromTAFF (QString *filename*) [virtual]

Reads the content of a taff-file

virtual void TriDecorator::ruppertRefinement () [virtual]

Adds points to make the triangles better shaped (algorithm of ruppert)

virtual bool TriDecorator::saveToTAFF (QString *filename*) const [virtual]

Saves the content to a taff file

virtual void TriDecorator::setBreakEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the breaklines

virtual void TriDecorator::setEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the normal edges

virtual void TriDecorator::setForcedCrossBehaviour (Triangulation::forcedCrossBehaviour *b*) [virtual]

Sets the behaviour of the triangulation in case of crossing forced lines

virtual void TriDecorator::setForcedEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the forced edges

virtual void TriDecorator::setTriangleInterpolator (TriangleInterpolator * *interpolator*) [virtual]

Sets an interpolator object

virtual bool TriDecorator::swapEdge (double *x*, double *y*) `[virtual]`

Swaps the edge which is closest to the point with x and y coordinates (if this is possible)

B.3 DualEdgeTriangulation Class Reference

B.3.1 Description

DualEdgeTriangulation is an implementation of a triangulation class based on the dual edge data structure

B.3.2 Member Function Documentation

void DualEdgeTriangulation::addLine (Line3D * *line*, bool *breakline*) `[virtual]`

Adds a line (e.g. a break-, structure- or an isoline) to the triangulation

int DualEdgeTriangulation::addPoint (Point3D * *p*) `[virtual]`

Adds a point to the triangulation and returns the number of this point in case of success or -100 in case of failure

int DualEdgeTriangulation::baseEdgeOfPoint (int *point*) `[protected]`

Returns the number of an edge which points to the point with number 'point' or -1 if there is an error

int DualEdgeTriangulation::baseEdgeOfTriangle (Point3D * *point*) `[protected]`

returns the number of a HalfEdge from a triangle in which 'point' is in. If the number -10 is returned, this means, that 'point' is outside the convex hull. If -5 is returned, then numerical problems with the leftOfTest occured (and the value of the possible edge is stored in the variable 'mUnstableEdge'. -20 means, that the inserted point is exactly on an edge (the number is stored in the variable 'mEdgeWithPoint'). -25 means, that the point is already in the triangulation (the number of the point is stored in the member 'mTwiceInsPoint'. If -100 is returned, this means that something else went wrong

virtual bool DualEdgeTriangulation::calcNormal (double *x*, double *y*, Vector3D * *result*) `[virtual]`

Calculates the normal at a point on the surface

virtual bool DualEdgeTriangulation::calcPoint (double *x*, double *y*, Point3D * *result*) `[virtual]`

Calculates x-, y and z-value of the point on the surface

bool DualEdgeTriangulation::checkSwap (unsigned int *edge*) `[protected]`

Checks, if 'edge' has to be swapped because of the empty circle criterion. If so, doSwap(...) is called.

void DualEdgeTriangulation::doOnlySwap (unsigned int *edge*) [protected]

Swaps 'edge' and does no recursiv testing

void DualEdgeTriangulation::doSwap (unsigned int *edge*) [protected]

Swaps 'edge' and test recursively for other swaps (delaunay criterion)

virtual void DualEdgeTriangulation::draw (QPainter * *p*, double *xlowleft*, double *ylowleft*, double *xupright*, double *yupright*, double *width*, double *height*) const [virtual]

draws the points, edges and the forced lines

bool DualEdgeTriangulation::edgeOnConvexHull (int *edge*) [protected]

Returns true, if a half edge is on the convex hull and false otherwise

void DualEdgeTriangulation::eliminateHorizontalTriangles () [virtual]

Eliminates the horizontal triangles by swapping or by insertion of new points

void DualEdgeTriangulation::evaluateInfluenceRegion (Point3D * *point*, int *edge*, std::set< int > * *set*) [protected]

Function needed for the ruppert algorithm. Tests, if point is in the circle through both endpoints of edge and the endpoint of edge->dual->next->point. If so, the function calls itself recursively for edge->next and edge->next->next. Stops, if it finds a forced edge or a convex hull edge

int DualEdgeTriangulation::getNumberOfPoints () [inline, virtual]

Returns the number of points

int DualEdgeTriangulation::getOppositePoint (int *p1*, int *p2*) [virtual]

Returns the number of the point opposite to the triangle points p1, p2 (which have to be on a halfedge)

Point3D * DualEdgeTriangulation::getPoint (unsigned int *i*) const [inline, virtual]

Returns a pointer to the point with number i

virtual QValueList<int>* DualEdgeTriangulation::getPointsAroundEdge (double *x*, double *y*) [virtual]

Returns a value list with the numbers of the four points, which would be affected by an edge swap. This function is e.g. needed by **NormVecDecorator** (p. **??**) to know the points, for which the normals have to be recalculated. The returned ValueList has to be deleted by the code which calls the method

QValueList<int>* DualEdgeTriangulation::getSurroundingTriangles (int *pointno*) `[virtual]`

Returns a pointer to a value list with the information of the triangles surrounding (counterclockwise) a point. Four integer values describe a triangle, the first three are the number of the half edges of the triangle and the fourth is -10, if the third (and most counterclockwise) edge is a breakline, and -20 otherwise. The value list has to be deleted by the code which called the method

virtual bool DualEdgeTriangulation::getTriangle (double *x*, double *y*, Point3D * *p1*, Point3D * *p2*, Point3D * *p3*) `[virtual]`

Finds out, in which triangle the point with coordinates x and y is and assigns adresses to the points at the vertices to 'p1', 'p2' and 'p3

virtual bool DualEdgeTriangulation::getTriangle (double *x*, double *y*, Point3D * *p1*, int * *n1*, Point3D * *p2*, int * *n2*, Point3D * *p3*, int * *n3*) `[virtual]`

Finds out, in which triangle the point with coordinates x and y is and assigns the numbers of the vertices to 'n1', 'n2' and 'n3' and the vertices to 'p1', 'p2' and 'p3'

double DualEdgeTriangulation::getXMax () `[inline, virtual]`

Returns the largest x-coordinate value of the bounding box

double DualEdgeTriangulation::getXMin () `[inline, virtual]`

Returns the smallest x-coordinate value of the bounding box

double DualEdgeTriangulation::getYMax () `[inline, virtual]`

Returns the largest y-coordinate value of the bounding box

double DualEdgeTriangulation::getYMin () `[inline, virtual]`

Returns the smallest x-coordinate value of the bounding box

bool DualEdgeTriangulation::halfEdgeBBoxTest (int *edge*, double *xlowleft*, double *ylowleft*, double *xupright*, double *yupright*) const `[inline, protected]`

Tests, if the bounding box of the halfedge with index i intersects the specified bounding box. The main purpose for this method is the drawing of the triangulation

unsigned int DualEdgeTriangulation::insertEdge (int *dual*, int *next*, int *point*, bool *mbreak*, bool *forced*) `[protected]`

inserts an edge and makes sure, everything is ok with the storage of the edge. The number of the HalfEdge is returned

int DualEdgeTriangulation::insertForcedSegment (int *p1*, int *p2*, bool *breakline*) [protected]

inserts a forced segment between the points with the numbers p1 and p2 into the triangulation and returns the number of a HalfEdge belonging to this forced edge or -100 in case of failure

virtual void DualEdgeTriangulation::performConsistencyTest () [virtual]

Performs a consistency check, remove this later

bool DualEdgeTriangulation::pointInside (double *x*, double *y*) [virtual]

Returns true, if the point with coordinates x and y is inside the convex hull and false otherwise

bool DualEdgeTriangulation::readFromTAFF (QString *filename*) [virtual]

Reads the dual edge structure of a taff file

void DualEdgeTriangulation::removeLine (int *i*)

Removes the line with number i from the triangulation

void DualEdgeTriangulation::removePoint (int *i*)

Removes the point with the number i from the triangulation

virtual void DualEdgeTriangulation::ruppertRefinement () [virtual]

Adds points to make the triangles better shaped (algorithm of ruppert)

bool DualEdgeTriangulation::saveToTAFF (QString *filename*) const [virtual]

Saves the dual edge structure to a taff file

virtual void DualEdgeTriangulation::setBreakEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the breaklines

virtual void DualEdgeTriangulation::setEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the normal edges

virtual void DualEdgeTriangulation::setForcedCrossBehaviour (Triangulation::forcedCross-Behaviour *b*) [virtual]

Sets the behaviour of the triangulation in case of crossing forced lines

virtual void DualEdgeTriangulation::setForcedEdgeColor (int *r*, int *g*, int *b*) [virtual]

Sets the color of the forced edges

void DualEdgeTriangulation::setTriangleInterpolator (TriangleInterpolator * *interpolator*) `[virtual]`

Sets an interpolator object

int DualEdgeTriangulation::splitHalfEdge (int *edge*, float *position*) `[protected]`

Inserts a new point on the halfedge with number 'edge'. The position can have a value from 0 to 1 (e.g. 0.5 would be in the middle). The return value is the number of the new inserted point. tin is the triangulation, which should be used to calculate the elevation of the inserted point

virtual bool DualEdgeTriangulation::swapEdge (double *x*, double *y*) `[virtual]`

Swaps the edge which is closest to the point with x and y coordinates (if this is possible)

double DualEdgeTriangulation::swapMinAngle (int *edge*) const `[protected]`

Calculates the minimum angle, which would be present, if the specified halfedge would be swapped

bool DualEdgeTriangulation::swapPossible (unsigned int *edge*) `[protected]`

Returns true, if it is possible to swap an edge, otherwise false(concave quad or edge on (or outside) the convex hull)

void DualEdgeTriangulation::triangulatePolygon (QValueList< int > * *poly*, QValueList< int > * *free*, int *mainedge*) `[protected]`

divides a polygon in a triangle and two polygons and calls itself recursively for these two polygons. 'poly' is a pointer to a list with the numbers of the edges of the polygon, 'free' is a pointer to a list of free halfedges, and 'mainedge' is the number of the edge, towards which the new triangle is inserted. Mainedge has to be the same as poly->begin(), otherwise the recursion does not work

B.3.3 Member Data Documentation

const double DualEdgeTriangulation::leftOfTresh = 0.00001 `[static, protected]`

Treshold for the leftOfTest to handle numerical instabilities

QColor DualEdgeTriangulation::mBreakEdgeColor `[protected]`

Color to paint the breaklines

Triangulation* DualEdgeTriangulation::mDecorator `[protected]`

Pointer to the decorator using this triangulation. It it is used directly, mDecorator equals this

const unsigned int DualEdgeTriangulation::mDefaultStorageForHalfEdges = 300006 `[static, protected]`

Default value for the number of storable HalfEdges at the beginning

const unsigned int DualEdgeTriangulation::mDefaultStorageForPoints = 50000 [static, protected]

Default value for the number of storable points at the beginning

QColor DualEdgeTriangulation::mEdgeColor [protected]

Color to paint the normal edges

unsigned int DualEdgeTriangulation::mEdgeInside [protected]

Number of an edge which does not point to the virtual point. It continuously updated for a fast search

unsigned int DualEdgeTriangulation::mEdgeOutside [protected]

Number of an edge on the outside of the convex hull. It is updated in method 'baseEdgeOfTriangle' to enable insertion of points outside the convex hull

unsigned int DualEdgeTriangulation::mEdgeWithPoint [protected]

If an inserted point is exactly on an existing edge, 'baseEdgeOfTriangle' returns -20 and sets the variable 'mEdgeWithPoint'

Triangulation::forcedCrossBehaviour DualEdgeTriangulation::mForcedCrossBehaviour [protected]

Member to store the behaviour in case of crossing forced segments

QColor DualEdgeTriangulation::mForcedEdgeColor [protected]

Color to paint the forced edges

QPtrVector<HalfEdge> DualEdgeTriangulation::mHalfEdge [protected]

Stores pointers to the HalfEdges

QPtrVector<Point3D> DualEdgeTriangulation::mPointVector [protected]

Stores pointers to all points in the triangulations (including the points contained in the lines)

TriangleInterpolator* DualEdgeTriangulation::mTriangleInterpolator [protected]

Association to an interpolator object

int DualEdgeTriangulation::mTwiceInsPoint [protected]

If a point has been inserted twice, its number is stored in this member

unsigned int DualEdgeTriangulation::mUnstableEdge [protected]

If an instability occurs in 'baseEdgeOfTriangle', mUnstableEdge is set to the value of the current edge

const int DualEdgeTriangulation::nBaseOfRuns = 300000 [static, protected]

Security to prevent endless loops in 'baseEdgeOfTriangle'. It there are more iteration then this number, the point will not be inserted

double DualEdgeTriangulation::xMax [protected]

X-coordinate of the upper right corner of the bounding box

double DualEdgeTriangulation::xMin [protected]

X-coordinate of the lower left corner of the bounding box

double DualEdgeTriangulation::yMax [protected]

Y-coordinate of the upper right corner of the bounding box

double DualEdgeTriangulation::yMin [protected]

Y-coordinate of the lower left corner of the bounding box

B.4 HalfEdge Class Reference

B.4.1 Description

Represents a half edge for the dual edge data structure

HalfEdge::HalfEdge () [inline]

Default constructor. Values for mDual, mNext, mPoint are set to -10 which means that they are undefined

B.4.2 Member Function Documentation

bool HalfEdge::getBreak () const [inline]

Returns, whether the HalfEdge belongs to a break line or not

int HalfEdge::getDual () const [inline]

Returns the number of the dual HalfEdge

bool HalfEdge::getForced () const [inline]

Returns, whether the HalfEdge belongs to a constrained edge or not

int HalfEdge::getNext () const `[inline]`

Returns the number of the next HalfEdge

int HalfEdge::getPoint () const `[inline]`

Returns the number of the point at which this HalfEdge points

void HalfEdge::setBreak (bool *b*) `[inline]`

Sets the break flag

void HalfEdge::setDual (int *d*) `[inline]`

Sets the number of the dual HalfEdge

void HalfEdge::setForced (bool *f*) `[inline]`

Sets the forced flag

void HalfEdge::setNext (int *n*) `[inline]`

Sets the number of the next HalfEdge

void HalfEdge::setPoint (int *p*) `[inline]`

Sets the number of point at which this HalfEdge points

B.4.3 Member Data Documentation

bool HalfEdge::mBreak `[protected]`

True, if the HalfEdge belongs to a break line, false otherwise

int HalfEdge::mDual `[protected]`

Number of the dual HalfEdge

bool HalfEdge::mForced `[protected]`

True, if the HalfEdge belongs to a constrained edge, false otherwise

int HalfEdge::mNext `[protected]`

Number of the next HalfEdge

int HalfEdge::mPoint `[protected]`

Number of the point at which this HalfEdge points

B.5 Line3D Class Reference

B.5.1 Description

This class represents a line. It is implemented as a single directed linked list of nodes (Point3D). Attention: the points inserted in a line are not deleted from Line3D

B.5.2 Member Function Documentation

bool Line3D::empty ()

returns true, if the Line contains no **Point3D** (p. **??**), otherwise false

unsigned int Line3D::getCurrent () `[inline]`

returns the current position

Point3D ∗ **Line3D::getPoint ()** `[inline]`

gets the point at the current position

unsigned int Line3D::getSize () `[inline]`

returns the size of the line (the numbero of inserted Nodes without 'head' and 'z'

void Line3D::goToBegin ()

sets the current Node

void Line3D::goToNext ()

goes to the next Node
inserts a node behind the current position and sets the current position to this new node

void Line3D::removePoint ()

removes the point behind the current position

B.6 Node Class Reference

B.6.1 Description

Node is a class used by Line3D. It represents a node in the single directed linked list. Associated Point3D objects are deleted when the node is deleted.

B.6.2 Member Function Documentation

Node ∗ **Node::getNext ()** `[inline]`

Returns a pointer to the next element in the linked list

Point3D ∗ Node::getPoint () `[inline]`

Returns a pointer to the Point3D object associated with the node

void Node::setNext (Node ∗ *n*) `[inline]`

Sets the pointer to the next node

void Node::setPoint (Point3D ∗ *p*) `[inline]`

Sets a new pointer to an associated Point3D object

B.6.3 Member Data Documentation

Node∗ Node::mNext `[protected]`

Pointer to the next Node in the linked list

Point3D∗ Node::mPoint `[protected]`

Pointer to the Point3D object associated with the node

B.7 TriangleInterpolator Class Reference

B.7.1 Description

This is an interface for interpolator classes for triangulations

B.7.2 Member Function Documentation

virtual bool TriangleInterpolator::calcNormVec (double *x*, double *y*, Vector3D ∗ *result*) `[pure virtual]`

Calculates the normal vector and assigns it to vec

virtual bool TriangleInterpolator::calcPoint (double *x*, double *y*, Point3D ∗ *result*) `[pure virtual]`

Performs an interpolation and assigns the x-,y- and z-coordinates to result

B.8 LinTriangleInterpolator Class Reference

B.8.1 Description

LinTriangleInterpolator is a class which interpolates linearly on a triangulation

B.8.2 Member Function Documentation

virtual bool LinTriangleInterpolator::calcFirstDerX (double *x*, double *y*, Vector3D * *result*) `[protected, virtual]`

Calculates the first derivative with respect to x for a linear surface and assigns it to vec

virtual bool LinTriangleInterpolator::calcFirstDerY (double *x*, double *y*, Vector3D * *result*) `[protected, virtual]`

Calculates the first derivative with respect to y for a linear surface and assigns it to vec

virtual bool LinTriangleInterpolator::calcNormVec (double *x*, double *y*, Vector3D * *result*) `[virtual]`

Calculates the normal vector and assigns it to vec

virtual bool LinTriangleInterpolator::calcPoint (double *x*, double *y*, Point3D * *result*) `[virtual]`

Performs a linear interpolation in a triangle and assigns the x-,y- and z-coordinates to point

DualEdgeTriangulation * LinTriangleInterpolator::getTriangulation () const `[inline, virtual]`

Returns a pointer to the current Triangulation object

void LinTriangleInterpolator::setTriangulation (DualEdgeTriangulation * *tin*) `[inline, virtual]`

Sets a Triangulation

B.9 CoonsTriangleInterpolator Class Reference

B.9.1 Detailed Description

This class interpolates a Coons-Patch-surface for a triangulation. In the current version, the cross-derivatives of the boundary curves are assessed using the x- and y-derivatives of the three nodes. They then are linear interpolated along each side (this can be changed in the methode 'predictCrossDer'). The ruled surface are blended using cubic Hermite interpolation.

B.9.2 Member Function Documentation

void CoonsTriangleInterpolator::calcFDPFDQRS2 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the mixed first derivative with respect to p and q on the ruled surface 2 in the standard triangle

void CoonsTriangleInterpolator::calcFirstDerPCorrTerm (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative of the correction term with respect to p

void CoonsTriangleInterpolator::calcFirstDerPRS1 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to p on the ruled surface 1 in the standard triangle. The second part is commented out at the moment

void CoonsTriangleInterpolator::calcFirstDerPRS12 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to p on the ruled surface 12 in the standard triangle. The second part is commented out at the moment

void CoonsTriangleInterpolator::calcFirstDerPRS2 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to p on the ruled surface 2 in the standard triangle

void CoonsTriangleInterpolator::calcFirstDerQCorrTerm (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative of the correction term with respect to q. the second part is commented out

void CoonsTriangleInterpolator::calcFirstDerQRS1 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to q on the ruled surface 1 in the standard triangle. The second part is commented out at the moment

void CoonsTriangleInterpolator::calcFirstDerQRS12 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to q on the ruled surface 2 in the standard triangle. The second part is commented out at the moment

void CoonsTriangleInterpolator::calcFirstDerQRS2 (double *p*, double *q*, Vector3D * *vec*) `[protected]`

calculates the first derivative with respect to q on the ruled surface 2 in the standard triangle. The second part is commented out

virtual bool CoonsTriangleInterpolator::calcNormVec (double *x*, double *y*, Vector3D * *vec*) `[virtual]`

Calculates the normal vector and assigns it to vec

virtual bool CoonsTriangleInterpolator::calcPoint (double *x*, double *y*, Point3D * *point*) `[virtual]`

Performs a Coons interpolation in a triangle and assigns the x-,y- and z-coordinates to point

void CoonsTriangleInterpolator::calcPointCorrTerm (double *p*, double *q*, Point3D * *point*) `[protected]`

calculates a point on the correction surface

void CoonsTriangleInterpolator::calcPointRS1 (double *p*, double *q*, Point3D * *point*) `[protected]`

calculates a point on the ruled surface1 with p- and q-coordinates in the standard triangle. Note that for the sake of speed only the first part of the ruled surface 1 is computed (the second part is commented out at the moment). The second part disappears when combining the ruled surfaces because of the second part of the ruled surface 12

void CoonsTriangleInterpolator::calcPointRS12 (double *p*, double *q*, Point3D * *point*) `[protected]`

calculates a point on the ruled surface12, which means that the ruled surface1 operator is applied to the result of the ruled surface 2 operator. As in the case of RS1, the second part is commented out because it disappears when combining all ruled surfaces to the Coons patch.

void CoonsTriangleInterpolator::calcPointRS2 (double *p*, double *q*, Point3D * *point*) `[protected]`

calculates a point on the ruled surface2 with p- and q-coordinates in the standard triangle

void CoonsTriangleInterpolator::init (double *x*, double *y*)

Finds out, in which triangle the point with coordinates is and assigns p, q, z-coordinates for mPoint, p1, p2, p3, p11, p12, p21, p22, p31, p32

void CoonsTriangleInterpolator::predictCrossDer (Vector3D * *begin*, int *numbegin*, Vector3D * *end*, int *numend*, float *param*, Vector3D * *result*) `[protected]`

Predicts a cross derivative at an edge according to the derivatives at the vertices and the position on the edge. 'begin' is the derivative at the first vertex, 'end' that of the second vertex. 'numbegin' is the number of the first vertex(1,2,or 3), 'numend' that of the second. The result will be assigned to 'result'.

void CoonsTriangleInterpolator::predictCrossDerDer (Vector3D * *begin*, int *numbegin*, Vector3D * *end*, int *numend*, float *param*, Vector3D * *result*) `[protected]`

Calculates the derivative of the CrossDerivative with respect to the parameter 'param'. The arguments have the same meaning as in 'predictCrossDer'

void CoonsTriangleInterpolator::transformPoint (Point3D * *thepoint,* double *x,* double *y*) `[protected]`

Transforms x-, y-, z-coordinates of a Point to p, q, z-coordinates. The standard triangle is defined by p1, p2, p3, so init should be run first to update p1, p2, p3

void CoonsTriangleInterpolator::transformPointBack (Point3D * *thepoint,* double *p,* double *q*) `[protected]`

Transforms p-, q-, z-coordinates of a Point to x,y,z-coordinates.

B.9.3 Member Data Documentation

QPtrVector<Point3D>* CoonsTriangleInterpolator::cpoly1 `[protected]`

Control polygon of pl1

QPtrVector<Point3D>* CoonsTriangleInterpolator::cpoly2 `[protected]`

Control polygon of pl2

QPtrVector<Point3D>* CoonsTriangleInterpolator::cpoly3 `[protected]`

Control polygon of pl3

Point3D CoonsTriangleInterpolator::d11 `[protected]`

Endpoint of the cross-derivative of pl1 at point 3

Point3D CoonsTriangleInterpolator::d12 `[protected]`

Endpoint of the cross-derivative of pl1 at point 2

Point3D CoonsTriangleInterpolator::d21 `[protected]`

Endpoint of the cross-derivative of pl2 at point 3

Point3D CoonsTriangleInterpolator::d22 `[protected]`

Endpoint of the cross-derivative of pl2 at point 1

Point3D CoonsTriangleInterpolator::d31 `[protected]`

Endpoint of the cross-derivative of pl3 at point 2

Point3D CoonsTriangleInterpolator::d32 `[protected]`

Endpoint of the cross-derivative of pl3 at point 1

Point3D CoonsTriangleInterpolator::fcp1 [protected]

First control point of pl1

Point3D CoonsTriangleInterpolator::fcp2 [protected]

First control point of pl2

Point3D CoonsTriangleInterpolator::fcp3 [protected]

First control point of pl3

Point3D CoonsTriangleInterpolator::focp1 [protected]

Fourth control point of pl1

Point3D CoonsTriangleInterpolator::focp2 [protected]

Fourth control point of pl2

Point3D CoonsTriangleInterpolator::focp3 [protected]

Fourth control point of pl3

Point3D CoonsTriangleInterpolator::lpoint1 [protected]

Caches the last point1 to check, if init has to be run again or not

Point3D CoonsTriangleInterpolator::lpoint2 [protected]

Caches the last point2 to check, if init has to be run again or not

Point3D CoonsTriangleInterpolator::lpoint3 [protected]

Caches the last point3 to check, if init has to be run again or not

Point3D CoonsTriangleInterpolator::mPoint [protected]

point(x,y) in coordinates p, q, z of the standard triangle

NormVecDecorator∗ CoonsTriangleInterpolator::mTIN [protected]

association with a triangulation object

Point3D CoonsTriangleInterpolator::p1 [protected]

first point of the triangle (p= 0, q=1 in the standard triangle)

Point3D CoonsTriangleInterpolator::p2 [protected]

second point of the triangle (p=1, q=0 in the standard triangle)

Point3D CoonsTriangleInterpolator::p3 [protected]

third point of the triangle (p=0, q=0 in the standard triangle)

Bezier3D CoonsTriangleInterpolator::pl1 [protected]

Parametric line opposite point 1 in p-, q- space

Bezier3D CoonsTriangleInterpolator::pl2 [protected]

Parametric line opposite point 2 in p-, q- space

Bezier3D CoonsTriangleInterpolator::pl3 [protected]

Parametric line opposite point 3 in p-, q- space

Point3D CoonsTriangleInterpolator::point1 [protected]

first point of the triangle in x-,y-,z-coordinates

Point3D CoonsTriangleInterpolator::point2 [protected]

second point of the triangle in x-,y-,z-coordinates

Point3D CoonsTriangleInterpolator::point3 [protected]

third point of the triangle in x-,y-,z-coordinates

Point3D CoonsTriangleInterpolator::scp1 [protected]

Second control point of pl1

Point3D CoonsTriangleInterpolator::scp2 [protected]

Second control point of pl2

Point3D CoonsTriangleInterpolator::scp3 [protected]

Second control point of pl3

Point3D CoonsTriangleInterpolator::tcp1 [protected]

Third control point of pl1

Point3D CoonsTriangleInterpolator::tcp2 [protected]

Third control point of pl2

Point3D CoonsTriangleInterpolator::tcp3 [protected]

Third control point of pl3

B.10 CloughTocherInterpolator Class Reference

B.10.1 Description

This is an implementation of a Clough-Tocher interpolator based on a triangular tessellation. The derivatives orthogonal to the boundary curves are interpolated linearly along a triangle edge.

B.10.2 Member Function Documentation

double CloughTocherInterpolator::calcBernsteinPoly (int *n*, int *i*, int *j*, int *k*, double *u*, double *v*, double *w*) `[protected]`

Calculates the Bernsteinpolynomials to calculate the Beziertriangle. 'n' is three in the cubical case, 'i', 'j', 'k' are the indices of the controllpoint and 'u', 'v', 'w' are the barycentric coordinates of the point

virtual bool CloughTocherInterpolator::calcNormVec (double *x*, double *y*, Vector3D * *result*) `[virtual]`

Calculates the normal vector and assigns it to vec (not implemented at the moment)

virtual bool CloughTocherInterpolator::calcPoint (double *x*, double *y*, Point3D * *result*) `[virtual]`

Performs a Clough-Tocher interpolation in a triangle and assigns the x-,y- and z-coordinates to result

void CloughTocherInterpolator::init (double *x*, double *y*) `[protected]`

Finds out, in which triangle the point with the coordinates x and y is and sets the elevation of the controlpoints

B.10.3 Member Data Documentation

double CloughTocherInterpolator::der1X `[protected]`

derivative in x-direction at point1

double CloughTocherInterpolator::der1Y `[protected]`

derivative in y-direction at point1

double CloughTocherInterpolator::der2X `[protected]`

derivative in x-direction at point2

double CloughTocherInterpolator::der2Y `[protected]`

derivative in y-direction at point2

double CloughTocherInterpolator::der3X [protected]

derivative in x-direction at point3

double CloughTocherInterpolator::der3Y [protected]

derivative in y-direction at point3

Point3D CloughTocherInterpolator::lpoint1 [protected]

stores point1 of the last run

Point3D CloughTocherInterpolator::lpoint2 [protected]

stores point2 of the last run

Point3D CloughTocherInterpolator::lpoint3 [protected]

stores point3 of the last run

double CloughTocherInterpolator::mEdgeTolerance [protected]

Tolerance of the barycentric coordinates at the borders of the triangles (to prevent errors because of very small negativ baricentric coordinates)

NormVecDecorator∗ CloughTocherInterpolator::mTIN [protected]

association with a triangulation object

Point3D CloughTocherInterpolator::point1 [protected]

first point of the triangle in x-,y-,z-coordinates

Point3D CloughTocherInterpolator::point2 [protected]

second point of the triangle in x-,y-,z-coordinates

Point3D CloughTocherInterpolator::point3 [protected]

third point of the triangle in x-,y-,z-coordinates

B.11 SCloughTocherInterpolator Class Reference

B.11.1 Description

This class implements a smoothed Clough-Tocher interpolator. The derivatives across the three sides are changed to get close to C2-continuity

B.11.2 Member Function Documentation

double SCloughTocherInterpolator::calcBernsteinPoly (int *n*, int *i*, int *j*, int *k*, double *u*, double *v*, double *w*) `[protected]`

Calculates the Bernsteinpolynomials to calculate the Beziertriangle. 'n' is three in the cubical case, 'i', 'j', 'k' are the indices of the controllpoint and 'u', 'v', 'w' are the barycentric coordinates of the point

virtual bool SCloughTocherInterpolator::calcNormVec (double *x*, double *y*, Vector3D * *result*) `[virtual]`

Calculates the normal vector and assigns it to vec (not implemented at the moment)

virtual bool SCloughTocherInterpolator::calcPoint (double *x*, double *y*, Point3D * *result*) `[virtual]`

Performs a smoothed Clough-Tocher interpolation in a triangle and assigns the x-,y- and z-coordinates to result

void SCloughTocherInterpolator::init (double *x*, double *y*) `[protected]`

Finds out, in which triangle the point with the coordinates x and y is and calculates the elevation of all necessary control points

void SCloughTocherInterpolator::smooth (int *i*) `[protected]`

Applies the smoothing algorithmus i times

B.11.3 Member Data Documentation

double SCloughTocherInterpolator::der1X `[protected]`

derivative in x-direction at point1

double SCloughTocherInterpolator::der1Y `[protected]`

derivative in y-direction at point1

double SCloughTocherInterpolator::der2X `[protected]`

derivative in x-direction at point2

double SCloughTocherInterpolator::der2Y `[protected]`

derivative in y-direction at point2

double SCloughTocherInterpolator::der3X `[protected]`

derivative in x-direction at point3

double SCloughTocherInterpolator::der3Y [protected]

derivative in y-direction at point3

double SCloughTocherInterpolator::der_1X [protected]

derivative in x-direction at point_1

double SCloughTocherInterpolator::der_1Y [protected]

derivative in y-direction at point_1

double SCloughTocherInterpolator::der_2X [protected]

derivative in x-direction at point_2

double SCloughTocherInterpolator::der_2Y [protected]

derivative in y-direction at point_2

double SCloughTocherInterpolator::der_3X [protected]

derivative in x-direction at point_3

double SCloughTocherInterpolator::der_3Y [protected]

derivative in y-direction at point_3

Point3D SCloughTocherInterpolator::lpoint1 [protected]

stores point1 of the last run

Point3D SCloughTocherInterpolator::lpoint2 [protected]

stores point2 of the last run

Point3D SCloughTocherInterpolator::lpoint3 [protected]

stores point3 of the last run

double SCloughTocherInterpolator::mEdgeTolerance [protected]

Tolerance of the barycentric coordinates at the borders of the triangles (to prevent errors because of very small negativ baricentric coordinates)

NormVecDecorator∗ SCloughTocherInterpolator::mTIN [protected]

association with a triangulation object

int SCloughTocherInterpolator::ptn1 [protected]

number of point1

int SCloughTocherInterpolator::ptn2 [protected]

number of point2

int SCloughTocherInterpolator::ptn3 [protected]

number of point3

int SCloughTocherInterpolator::ptn_1 [protected]

number of point_1

int SCloughTocherInterpolator::ptn_2 [protected]

number of point_2

int SCloughTocherInterpolator::ptn_3 [protected]

number of point_3

NormVecDecorator::pointState SCloughTocherInterpolator::state1 [protected]

state of point1 (NORMAL, BREAKLINE, ENDPOINT possible)

NormVecDecorator::pointState SCloughTocherInterpolator::state2 [protected]

state of point1 (NORMAL, BREAKLINE, ENDPOINT possible)

NormVecDecorator::pointState SCloughTocherInterpolator::state3 [protected]

state of point1 (NORMAL, BREAKLINE, ENDPOINT possible)

NormVecDecorator::pointState SCloughTocherInterpolator::state_1 [protected]

state of point_1 (NORMAL, BREAKLINE, ENDPOINT possible)

NormVecDecorator::pointState SCloughTocherInterpolator::state_2 [protected]

state of point_2 (NORMAL, BREAKLINE, ENDPOINT possible)

NormVecDecorator::pointState SCloughTocherInterpolator::state_3 [protected]

state of point_3 (NORMAL, BREAKLINE, ENDPOINT possible)

Curriculum vitae

Marco Hugentobler
born October 27th, 1975, in Mönchaltorf (ZH)

Schooling

1982 - 1988 Primary school in Mönchaltorf
1988 - 1994 Kantonsschule Zürcher Oberland in Wetzikon, Matura type E
1995 - 2000 MSc in Geography at the University of Zurich
with a thesis on 'Fortpflanzung von Unsicherheiten in dreiecksbasierten digitalen Geländemodellen mit Intervallarithmetik'
('Propagation of uncertainties in triangular digital terrain models with interval arithmetics')
advised by Prof. Dr. Robert Weibel and Dr. Bernhard Schneider
2000 - 2004 PhD at the Department of Geography of the University of Zürich
with a thesis on 'Terrain Modelling with Triangle based Free-Form Surfaces'
advised by Prof. Dr. Robert Weibel, Dr. Ross Purves and Dr. Bernhard Schneider

Publications

- Hugentobler, M. (2000). Fortpflanzung von Unsicherheiten in dreiecksbasierten digitalen Geländemodellen mit Intervallarithmetik. Master's thesis, Department of Geography, University of Zurich.
- Hugentobler, M. (2001). Propagation of uncertainties in digital terrain models with interval methods. In *Proceedings of the GIS Research UK 9th Annual Conference*, pages 341-344.
- Hugentobler, M. (2002). Interpolation of continuous surfaces for terrain modelling with Coons patches. In *Proceedings of the GISRUK 2002*, pages 13-15.
- Hugentobler, M. and Schneider, B. (2004). Breaklines in Coons surfaces over triangles for the use in terrain modelling. Submitted to Computers & Geosciences.
- Hugentobler, M., Purves, R.and Schneider, B. (2004). Evaluating methods for interpolating continuous surfaces from irregular data: a case study. Accepted as conference paper for Spatial Data Handling 2004.